

MergeSortProject

Wygenerowano za pomocą Doxygen 1.15.0

Rozdział 1

Struktura katalogów

1.1 Katalogi

build	??
native	??
include	??
gtest	??
internal	??
custom	??
gtest-port.h	??
gtest-printers.h	??
gtest.h	??
gtest-death-test-internal.h	??
gtest-filepath.h	??
gtest-internal.h	??
gtest-linked_ptr.h	??
gtest-param-util-generated.h	??
gtest-param-util.h	??
gtest-port-arch.h	??
gtest-port.h	??
gtest-string.h	??
gtest-tuple.h	??
gtest-type-util.h	??
gtest-death-test.h	??
gtest-message.h	??
gtest-param-test.h	??
gtest-printers.h	??
gtest-spi.h	??
gtest-test-part.h	??
gtest-typed-test.h	??
gtest.h	??
gtest_pred_impl.h	??
gtest_prod.h	??
custom	??
gtest-port.h	??
gtest-printers.h	??
gtest.h	??
gtest	??
internal	??
custom	??

gtest-port.h	??
gtest-printers.h	??
gtest.h	??
gtest-death-test-internal.h	??
gtest-filename.h	??
gtest-internal.h	??
gtest-linked_ptr.h	??
gtest-param-util-generated.h	??
gtest-param-util.h	??
gtest-port-arch.h	??
gtest-port.h	??
gtest-string.h	??
gtest-tuple.h	??
gtest-type-util.h	??
gtest-death-test.h	??
gtest-message.h	??
gtest-param-test.h	??
gtest-printers.h	??
gtest-spi.h	??
gtest-test-part.h	??
gtest-typed-test.h	??
gtest.h	??
gtest_pred_impl.h	??
gtest_prod.h	??
include	??
gtest	??
internal	??
custom	??
gtest-port.h	??
gtest-printers.h	??
gtest.h	??
gtest-death-test-internal.h	??
gtest-filename.h	??
gtest-internal.h	??
gtest-linked_ptr.h	??
gtest-param-util-generated.h	??
gtest-param-util.h	??
gtest-port-arch.h	??
gtest-port.h	??
gtest-string.h	??
gtest-tuple.h	??
gtest-type-util.h	??
gtest-death-test.h	??
gtest-message.h	??
gtest-param-test.h	??
gtest-printers.h	??
gtest-spi.h	??
gtest-test-part.h	??
gtest-typed-test.h	??
gtest.h	??
gtest_pred_impl.h	??
gtest_prod.h	??
internal	??
custom	??
gtest-port.h	??
gtest-printers.h	??
gtest.h	??

gtest-death-test-internal.h	??
gtest-filepath.h	??
gtest-internal.h	??
gtest-linked_ptr.h	??
gtest-param-util-generated.h	??
gtest-param-util.h	??
gtest-port-arch.h	??
gtest-port.h	??
gtest-string.h	??
gtest-tuple.h	??
gtest-type-util.h	??
MergeSortApp	??
MergeSort.h	??
MergeSortApp.cpp	??
Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8	??
build	??
native	??
include	??
gtest	??
internal	??
custom	??
gtest-port.h	??
gtest-printers.h	??
gtest.h	??
gtest-death-test-internal.h	??
gtest-filepath.h	??
gtest-internal.h	??
gtest-linked_ptr.h	??
gtest-param-util-generated.h	??
gtest-param-util.h	??
gtest-port-arch.h	??
gtest-port.h	??
gtest-string.h	??
gtest-tuple.h	??
gtest-type-util.h	??
gtest-death-test.h	??
gtest-message.h	??
gtest-param-test.h	??
gtest-printers.h	??
gtest-spi.h	??
gtest-test-part.h	??
gtest-typed-test.h	??
gtest.h	??
gtest_pred_impl.h	??
gtest_prod.h	??
native	??
include	??
gtest	??
internal	??
custom	??
gtest-port.h	??
gtest-printers.h	??
gtest.h	??
gtest-death-test-internal.h	??
gtest-filepath.h	??
gtest-internal.h	??
gtest-linked_ptr.h	??

gtest-param-util-generated.h	??
gtest-param-util.h	??
gtest-port-arch.h	??
gtest-port.h	??
gtest-string.h	??
gtest-tuple.h	??
gtest-type-util.h	??
gtest-death-test.h	??
gtest-message.h	??
gtest-param-test.h	??
gtest-printers.h	??
gtest-spi.h	??
gtest-test-part.h	??
gtest-typed-test.h	??
gtest.h	??
gtest_pred_impl.h	??
gtest_prod.h	??
packages	??
Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8	??
build	??
native	??
include	??
gtest	??
internal	??
custom	??
gtest-port.h	??
gtest-printers.h	??
gtest.h	??
gtest-death-test-internal.h	??
gtest-filepath.h	??
gtest-internal.h	??
gtest-linked_ptr.h	??
gtest-param-util-generated.h	??
gtest-param-util.h	??
gtest-port-arch.h	??
gtest-port.h	??
gtest-string.h	??
gtest-tuple.h	??
gtest-type-util.h	??
gtest-death-test.h	??
gtest-message.h	??
gtest-param-test.h	??
gtest-printers.h	??
gtest-spi.h	??
gtest-test-part.h	??
gtest-typed-test.h	??
gtest.h	??
gtest_pred_impl.h	??
gtest_prod.h	??

Rozdział 2

Indeks przestrzeni nazw

2.1 Lista przestrzeni nazw

Tutaj znajdują się wszystkie przestrzenie nazw wraz z ich krótkimi opisami:

proto2	??
std	??
std::tr1	??
std::tr1::gtest_internal	??
testing	??
testing::internal	??
testing::internal2	??
testing::internal::edit_distance	??
testing::internal::posix	??
testing_internal	??

Rozdział 3

Indeks hierarchiczny

3.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

```
std::tr1::gtest_internal::AddRef< T > . . . . . ??  
std::tr1::gtest_internal::AddRef< T & > . . . . . ??  
testing::internal::AddReference< T > . . . . . ??  
testing::internal::AddReference< T & > . . . . . ??  
testing::internal::AssertHelper . . . . . ??  
testing::internal::bool_constant< bool_value > . . . . . ??  
testing::internal::bool_constant< false > . . . . . ??  
testing::internal::HasValueType< T, VoidT< typename T::value_type > > . . . . . ??  
testing::internal::is_pointer< T * > . . . . . ??  
testing::internal::is_same< value_type, C > . . . . . ??  
testing::internal::is_same< T, T > . . . . . ??  
testing::internal::HasValueType< T, typename > . . . . . ??  
testing::internal::IsRecursiveContainerImpl< C, false, HV > . . . . . ??  
testing::internal::IsRecursiveContainerImpl< C, true, false > . . . . . ??  
testing::internal::is_pointer< T > . . . . . ??  
testing::internal::is_same< T, U > . . . . . ??  
testing::internal::bool_constant< ImplicitlyConvertible< const T *, const ProtocolMessage * >value||←  
ImplicitlyConvertible< const T *, const proto2::Message * >value > . . . . . ??  
testing::internal::IsAProtocolMessage< T > . . . . . ??  
testing::internal::bool_constant< true > . . . . . ??  
testing::internal::HasValueType< T, VoidT< typename T::value_type > > . . . . . ??  
testing::internal::is_pointer< T * > . . . . . ??  
testing::internal::is_same< T, T > . . . . . ??  
std::tr1::gtest_internal::ByRef< T > . . . . . ??  
std::tr1::gtest_internal::ByRef< T & > . . . . . ??  
testing::internal::CodeLocation . . . . . ??  
testing::internal::CompileAssert< bool > . . . . . ??  
testing::internal::CompileAssertTypesEqual< T1, T2 > . . . . . ??  
testing::internal::CompileAssertTypesEqual< T, T > . . . . . ??  
testing::internal::ConstCharPtr . . . . . ??  
testing::internal::ConstRef< T > . . . . . ??  
testing::internal::ConstRef< T & > . . . . . ??  
testing::internal::EnableIf< bool > . . . . . ??  
testing::internal::EnableIf< true > . . . . . ??
```

testing::Environment	??
testing::internal::EqHelper< lhs_is_null_literal >	??
testing::internal::EqHelper< true >	??
testing::internal::FloatingPoint< RawType >	??
testing::internal::FormatForComparison< ToPrint, OtherOperand >	??
testing::internal::FormatForComparison< ToPrint[N], OtherOperand >	??
std::tr1::gtest_internal::Get< k >	??
std::tr1::gtest_internal::Get< 0 >	??
std::tr1::gtest_internal::Get< 1 >	??
std::tr1::gtest_internal::Get< 2 >	??
std::tr1::gtest_internal::Get< 3 >	??
std::tr1::gtest_internal::Get< 4 >	??
std::tr1::gtest_internal::Get< 5 >	??
std::tr1::gtest_internal::Get< 6 >	??
std::tr1::gtest_internal::Get< 7 >	??
std::tr1::gtest_internal::Get< 8 >	??
std::tr1::gtest_internal::Get< 9 >	??
testing::internal::GTestLog	??
testing::internal::GTestMutexLock	??
testing::internal::ImplicitlyConvertible< From, To >	??
testing::internal::IsHashTable< T >	??
testing::internal::IsRecursiveContainerImpl< C, bool, bool >	??
testing::internal::IsRecursiveContainerImpl< C, true, true >	??
testing::internal::IsSame< T, U >	??
testing::internal::IsSame< T, T >	??
testing::internal::IteratorTraits< Iterator >	??
testing::internal::IteratorTraits< const T * >	??
testing::internal::IteratorTraits< T * >	??
testing::internal::linked_ptr< T >	??
testing::internal::linked_ptr_internal	??
MergeSort< T >	??
testing::Message	??
testing::internal::Mutex	??
testing::internal::NativeArray< Element >	??
testing::internal::ParameterizedTestCaseInfoBase	??
testing::internal::ParameterizedTestCaseInfo< TestCase >	??
testing::internal::ParameterizedTestCaseRegistry	??
testing::internal::ParamGenerator< T >	??
testing::internal::ParamGeneratorInterface< T >	??
testing::internal::RangeGenerator< T, IncrementT >	??
testing::internal::ValuesInIteratorRangeGenerator< T >	??
testing::internal::ParamIterator< T >	??
testing::internal::ParamIteratorInterface< T >	??
testing::internal::ParamNameGenFunc< ParamType >	??
testing::PrintToStringParamName	??
testing::internal::Random	??
testing::internal::RE	??
testing::internal::RelationToSourceCopy	??
testing::internal::RelationToSourceReference	??
testing::internal::RemoveConst< T >	??
testing::internal::RemoveConst< const T >	??
testing::internal::RemoveConst< const T[N]>	??
testing::internal::RemoveReference< T >	??
testing::internal::RemoveReference< T & >	??
testing::internal::RvalueRef< T >	??
std::tr1::gtest_internal::SameSizeTuplePrefixComparator< kSize1, kSize2 >	??
std::tr1::gtest_internal::SameSizeTuplePrefixComparator< 0, 0 >	??
std::tr1::gtest_internal::SameSizeTuplePrefixComparator< k, k >	??

```
testing::internal::scoped_ptr< T > . . . . . ??  
testing::ScopedTrace . . . . . ??  
testing::internal::StaticAssertTypeEqHelper< T1, T2 > . . . . . ??  
testing::internal::StaticAssertTypeEqHelper< T, T > . . . . . ??  
testing::internal::String . . . . . ??  
testing::Test . . . . . ??  
    testing::TestWithParam< T > . . . . . ??  
testing::TestCase . . . . . ??  
testing::TestEventListener . . . . . ??  
    testing::EmptyTestEventListener . . . . . ??  
testing::TestEventListeners . . . . . ??  
testing::internal::TestFactoryBase . . . . . ??  
    testing::internal::ParameterizedTestFactory< TestClass > . . . . . ??  
    testing::internal::TestFactoryImpl< TestClass > . . . . . ??  
testing::TestInfo . . . . . ??  
testing::internal::TestMetaFactoryBase< ParamType > . . . . . ??  
testing::internal::TestMetaFactoryBase< TestCase::ParamType > . . . . . ??  
    testing::internal::TestMetaFactory< TestCase > . . . . . ??  
testing::TestParamInfo< ParamType > . . . . . ??  
testing::TestProperty . . . . . ??  
testing::TestResult . . . . . ??  
testing::internal::ThreadLocal< T > . . . . . ??  
std::tr1::tuple<> . . . . . ??  
std::tr1::tuple<> . . . . . ??  
std::tr1::tuple_element< k, Tuple > . . . . . ??  
std::tr1::tuple_size< Tuple > . . . . . ??  
std::tr1::tuple_size< GTEST_0_TUPLE_(T) > . . . . . ??  
std::tr1::tuple_size< GTEST_10_TUPLE_(T) > . . . . . ??  
std::tr1::tuple_size< GTEST_1_TUPLE_(T) > . . . . . ??  
std::tr1::tuple_size< GTEST_2_TUPLE_(T) > . . . . . ??  
std::tr1::tuple_size< GTEST_3_TUPLE_(T) > . . . . . ??  
std::tr1::tuple_size< GTEST_4_TUPLE_(T) > . . . . . ??  
std::tr1::tuple_size< GTEST_5_TUPLE_(T) > . . . . . ??  
std::tr1::tuple_size< GTEST_6_TUPLE_(T) > . . . . . ??  
std::tr1::tuple_size< GTEST_7_TUPLE_(T) > . . . . . ??  
std::tr1::tuple_size< GTEST_8_TUPLE_(T) > . . . . . ??  
std::tr1::tuple_size< GTEST_9_TUPLE_(T) > . . . . . ??  
std::tr1::gtest_internal::TupleElement< kIndexValid, kIndex, Tuple > . . . . . ??  
std::tr1::gtest_internal::TupleElement< true, 0, GTEST_10_TUPLE_(T) > . . . . . ??  
std::tr1::gtest_internal::TupleElement< true, 1, GTEST_10_TUPLE_(T) > . . . . . ??  
std::tr1::gtest_internal::TupleElement< true, 2, GTEST_10_TUPLE_(T) > . . . . . ??  
std::tr1::gtest_internal::TupleElement< true, 3, GTEST_10_TUPLE_(T) > . . . . . ??  
std::tr1::gtest_internal::TupleElement< true, 4, GTEST_10_TUPLE_(T) > . . . . . ??  
std::tr1::gtest_internal::TupleElement< true, 5, GTEST_10_TUPLE_(T) > . . . . . ??  
std::tr1::gtest_internal::TupleElement< true, 6, GTEST_10_TUPLE_(T) > . . . . . ??  
std::tr1::gtest_internal::TupleElement< true, 7, GTEST_10_TUPLE_(T) > . . . . . ??  
std::tr1::gtest_internal::TupleElement< true, 8, GTEST_10_TUPLE_(T) > . . . . . ??  
std::tr1::gtest_internal::TupleElement< true, 9, GTEST_10_TUPLE_(T) > . . . . . ??  
testing::internal::TuplePolicy< TupleT > . . . . . ??  
IsRecursiveContainerImpl::type  
    testing::internal::IsRecursiveContainer< C > . . . . . ??  
testing::internal::TypeIdHelper< T > . . . . . ??  
testing::internal2::TypeWithoutFormatter< T, kTypeKind > . . . . . ??  
testing::internal2::TypeWithoutFormatter< T, kConvertibleToInteger > . . . . . ??  
testing::internal2::TypeWithoutFormatter< T, kProtobuf > . . . . . ??  
testing::internal::TypeWithSize< size > . . . . . ??  
testing::internal::TypeWithSize< 4 > . . . . . ??
```


Rozdział 4

Indeks klas

4.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

std::tr1::gtest_internal::AddRef< T >	??
std::tr1::gtest_internal::AddRef< T & >	??
testing::internal::AddReference< T >	??
testing::internal::AddReference< T & >	??
testing::internal::AssertHelper	??
testing::internal::bool_constant< bool_value >	??
std::tr1::gtest_internal::ByRef< T >	??
std::tr1::gtest_internal::ByRef< T & >	??
testing::internal::CodeLocation	??
testing::internal::CompileAssert< bool >	??
testing::internal::CompileAssertTypesEqual< T1, T2 >	??
testing::internal::CompileAssertTypesEqual< T, T >	??
testing::internal::ConstCharPtr	??
testing::internal::ConstRef< T >	??
testing::internal::ConstRef< T & >	??
testing::EmptyTestEventListener	??
testing::internal::EnableIf< bool >	??
testing::internal::EnableIf< true >	??
testing::Environment	??
testing::internal::EqHelper< lhs_is_null_literal >	??
testing::internal::EqHelper< true >	??
testing::internal::FloatingPoint< RawType >	??
testing::internal::FormatForComparison< ToPrint, OtherOperand >	??
testing::internal::FormatForComparison< ToPrint[N], OtherOperand >	??
std::tr1::gtest_internal::Get< k >	??
std::tr1::gtest_internal::Get< 0 >	??
std::tr1::gtest_internal::Get< 1 >	??
std::tr1::gtest_internal::Get< 2 >	??
std::tr1::gtest_internal::Get< 3 >	??
std::tr1::gtest_internal::Get< 4 >	??
std::tr1::gtest_internal::Get< 5 >	??
std::tr1::gtest_internal::Get< 6 >	??
std::tr1::gtest_internal::Get< 7 >	??
std::tr1::gtest_internal::Get< 8 >	??
std::tr1::gtest_internal::Get< 9 >	??

testing::internal::GTestLog	??
testing::internal::GTestMutexLock	??
testing::internal::HasValueType< T, typename >	??
testing::internal::HasValueType< T, VoidT< typename T::value_type > >	??
testing::internal::ImplicitlyConvertible< From, To >	??
testing::internal::is_pointer< T >	??
testing::internal::is_pointer< T * >	??
testing::internal::is_same< T, U >	??
testing::internal::is_same< T, T >	??
testing::internal::IsAProtocolMessage< T >	??
testing::internal::IsHashTable< T >	??
testing::internal::IsRecursiveContainer< C >	??
testing::internal::IsRecursiveContainerImpl< C, bool, bool >	??
testing::internal::IsRecursiveContainerImpl< C, false, HV >	??
testing::internal::IsRecursiveContainerImpl< C, true, false >	??
testing::internal::IsRecursiveContainerImpl< C, true, true >	??
testing::internal::IsSame< T, U >	??
testing::internal::IsSame< T, T >	??
testing::internal::IteratorTraits< Iterator >	??
testing::internal::IteratorTraits< const T * >	??
testing::internal::IteratorTraits< T * >	??
testing::internal::linked_ptr< T >	??
testing::internal::linked_ptr_internal	??
MergeSort< T >	
Klasa szablonowa realizująca algorytm sortowania przez scalanie	??
testing::Message	??
testing::internal::Mutex	??
testing::internal::NativeArray< Element >	??
testing::internal::ParameterizedTestCaseInfo< TestCase >	??
testing::internal::ParameterizedTestCaseInfoBase	??
testing::internal::ParameterizedTestCaseRegistry	??
testing::internal::ParameterizedTestFactory< TestClass >	??
testing::internal::ParamGenerator< T >	??
testing::internal::ParamGeneratorInterface< T >	??
testing::internal::ParamIterator< T >	??
testing::internal::ParamIteratorInterface< T >	??
testing::internal::ParamNameGenFunc< ParamType >	??
testing::PrintToStringParamName	??
testing::internal::Random	??
testing::internal::RangeGenerator< T, IncrementT >	??
testing::internal::RE	??
testing::internal::RelationToSourceCopy	??
testing::internal::RelationToSourceReference	??
testing::internal::RemoveConst< T >	??
testing::internal::RemoveConst< const T >	??
testing::internal::RemoveConst< const T[N]>	??
testing::internal::RemoveReference< T >	??
testing::internal::RemoveReference< T & >	??
testing::internal::RvalueRef< T >	??
std::tr1::gtest_internal::SameSizeTuplePrefixComparator< kSize1, kSize2 >	??
std::tr1::gtest_internal::SameSizeTuplePrefixComparator< 0, 0 >	??
std::tr1::gtest_internal::SameSizeTuplePrefixComparator< k, k >	??
testing::internal::scoped_ptr< T >	??
testing::ScopedTrace	??
testing::internal::StaticAssertTypeEqHelper< T1, T2 >	??
testing::internal::StaticAssertTypeEqHelper< T, T >	??
testing::internal::String	??
testing::Test	??

testing::TestCase ??
testing::TestEventListener ??
testing::TestEventListeners ??
testing::internal::TestFactoryBase ??
testing::internal::TestFactoryImpl< TestCase > ??
testing::TestInfo ??
testing::internal::TestMetaFactory< TestCase > ??
testing::internal::TestMetaFactoryBase< ParamType > ??
testing::TestParamInfo< ParamType > ??
testing::TestProperty ??
testing::TestResult ??
testing::TestWithParam< T > ??
testing::internal::ThreadLocal< T > ??
std::tr1::tuple<> ??
std::tr1::tuple<> ??
std::tr1::tuple_element< k, Tuple > ??
std::tr1::tuple_size< Tuple > ??
std::tr1::tuple_size< GTEST_0_TUPLE_(T) > ??
std::tr1::tuple_size< GTEST_10_TUPLE_(T) > ??
std::tr1::tuple_size< GTEST_1_TUPLE_(T) > ??
std::tr1::tuple_size< GTEST_2_TUPLE_(T) > ??
std::tr1::tuple_size< GTEST_3_TUPLE_(T) > ??
std::tr1::tuple_size< GTEST_4_TUPLE_(T) > ??
std::tr1::tuple_size< GTEST_5_TUPLE_(T) > ??
std::tr1::tuple_size< GTEST_6_TUPLE_(T) > ??
std::tr1::tuple_size< GTEST_7_TUPLE_(T) > ??
std::tr1::tuple_size< GTEST_8_TUPLE_(T) > ??
std::tr1::tuple_size< GTEST_9_TUPLE_(T) > ??
std::tr1::gtest_internal::TupleElement< kIndexValid, kIndex, Tuple > ??
std::tr1::gtest_internal::TupleElement< true, 0, GTEST_10_TUPLE_(T) > ??
std::tr1::gtest_internal::TupleElement< true, 1, GTEST_10_TUPLE_(T) > ??
std::tr1::gtest_internal::TupleElement< true, 2, GTEST_10_TUPLE_(T) > ??
std::tr1::gtest_internal::TupleElement< true, 3, GTEST_10_TUPLE_(T) > ??
std::tr1::gtest_internal::TupleElement< true, 4, GTEST_10_TUPLE_(T) > ??
std::tr1::gtest_internal::TupleElement< true, 5, GTEST_10_TUPLE_(T) > ??
std::tr1::gtest_internal::TupleElement< true, 6, GTEST_10_TUPLE_(T) > ??
std::tr1::gtest_internal::TupleElement< true, 7, GTEST_10_TUPLE_(T) > ??
std::tr1::gtest_internal::TupleElement< true, 8, GTEST_10_TUPLE_(T) > ??
std::tr1::gtest_internal::TupleElement< true, 9, GTEST_10_TUPLE_(T) > ??
testing::internal::TuplePolicy< TupleT > ??
testing::internal::TypeIdHelper< T > ??
testing::internal2::TypeWithoutFormatter< T, kTypeKind > ??
testing::internal2::TypeWithoutFormatter< T, kConvertibleToInteger > ??
testing::internal2::TypeWithoutFormatter< T, kProtobuf > ??
testing::internal::TypeWithSize< size > ??
testing::internal::TypeWithSize< 4 > ??
testing::internal::TypeWithSize< 8 > ??
testing::UnitTest ??
testing::internal::UniversalPrinter< T > ??
testing::internal::UniversalPrinter< T & > ??
testing::internal::UniversalPrinter< T[N]> ??
testing::internal::UniversalTersePrinter< T > ??
testing::internal::UniversalTersePrinter< char * > ??
testing::internal::UniversalTersePrinter< const char * > ??
testing::internal::UniversalTersePrinter< T & > ??
testing::internal::UniversalTersePrinter< T[N]> ??
testing::internal::UniversalTersePrinter< wchar_t * > ??
testing::internal::ValueArray1< T1 > ??


```
testing::internal::ValueArray40< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21,  
??  
testing::internal::ValueArray41< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21,  
??  
testing::internal::ValueArray42< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21,  
??  
testing::internal::ValueArray43< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21,  
??  
testing::internal::ValueArray44< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21,  
??  
testing::internal::ValueArray45< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21,  
??  
testing::internal::ValueArray46< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21,  
??  
testing::internal::ValueArray47< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21,  
??  
testing::internal::ValueArray48< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21,  
??  
testing::internal::ValueArray49< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21,  
??  
testing::internal::ValueArray5< T1, T2, T3, T4, T5 > . . . . . ??  
testing::internal::ValueArray50< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21,  
??  
testing::internal::ValueArray6< T1, T2, T3, T4, T5, T6 > . . . . . ??  
testing::internal::ValueArray7< T1, T2, T3, T4, T5, T6, T7 > . . . . . ??  
testing::internal::ValueArray8< T1, T2, T3, T4, T5, T6, T7, T8 > . . . . . ??  
testing::internal::ValueArray9< T1, T2, T3, T4, T5, T6, T7, T8, T9 > . . . . . ??  
testing::internal::ValuesInIteratorRangeGenerator< T > . . . . . ??  
testing::internal::VoidT< T > . . . . . ??  
testing::WithParamInterface< T > . . . . . ??  
testing::internal::WrapPrinterType< type > . . . . . ??
```


Rozdział 5

Indeks plików

5.1 Lista plików

Tutaj znajduje się lista wszystkich plików wraz z ich krótkimi opisami:

pch.cpp	??
pch.h	??
test.cpp	??
MergeSortApp/MergeSort.h	Implementacja algorytmu sortowania przez scalanie (Merge Sort)	??
MergeSortApp/MergeSortApp.cpp	Główny plik programu uruchamiający algorytm	??
packages/Microsoft.gunittest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-death-test.h	??	
packages/Microsoft.gunittest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-message.h	??	
packages/Microsoft.gunittest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-param-test.h	??	
packages/Microsoft.gunittest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h	??	
packages/Microsoft.gunittest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-spi.h	??	
packages/Microsoft.gunittest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-test-part.h	??	
packages/Microsoft.gunittest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-typed-test.h	??	
packages/Microsoft.gunittest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h	??	
packages/Microsoft.gunittest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest_pred_impl.h	??	
packages/Microsoft.gunittest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest_prod.h	??	
packages/Microsoft.gunittest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-death-test-int	??	
packages/Microsoft.gunittest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-filepath.h	??	
packages/Microsoft.gunittest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h	??	
packages/Microsoft.gunittest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-linked_ptr.h	??	

packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-param-util.h](#)
??
packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-param-util.h](#)
??
packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-port-arch.h](#)
??
packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-port.h](#)
??
packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-string.h](#)
??
packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-tuple.h](#)
??
packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-type-util.h](#)
??
packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/custom/[gtest-port.h](#)
??
packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/custom/[gtest-printer.h](#)
??
packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/custom/[gtest.h](#)
??

Rozdział 6

Dokumentacja katalogów

6.1 Dokumentacja katalogu packages/Microsoft.googletest.v140. \leftarrow windesktop.msvcstl.static.rt-dyn.1.8.1.8/build

Katalogi

- katalog [native](#)

6.2 Dokumentacja katalogu packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt- dyn.1.8.1.8/build/native/include/gtest/internal/custom

Pliki

- plik [gtest-port.h](#)
- plik [gtest-printers.h](#)
- plik [gtest.h](#)

6.2.1 Opis szczegółowy

6.2.2 Customization Points

The custom directory is an injection point for custom user configurations.

6.2.3 Header gtest.h

6.2.3.1 The following macros can be defined:

- GTEST_OS_STACK_TRACE_GETTER_ - The name of an implementation of [OsStackTraceGetter](#) Interface.
- GTEST_CUSTOM_TEMPDIR_FUNCTION_ - An override for [testing::TempDir\(\)](#). See [testing::TempDir](#) for semantics and signature.

6.2.4 Header `gtest-port.h`

The following macros can be defined:

6.2.4.1 Flag related macros:

- `GTEST_FLAG(flag_name)`
- `GTEST_USE_OWN_FLAGFILE_FLAG_` - Define to 0 when the system provides its own flagfile flag parsing.
- `GTEST_DECLARE_bool_(name)`
- `GTEST_DECLARE_int32_(name)`
- `GTEST_DECLARE_string_(name)`
- `GTEST_DEFINE_bool_(name, default_val, doc)`
- `GTEST_DEFINE_int32_(name, default_val, doc)`
- `GTEST_DEFINE_string_(name, default_val, doc)`

6.2.4.2 Logging:

- `GTEST_LOG_(severity)`
- `GTEST_CHECK_(condition)`
- Functions `LogToStderr()` and `FlushInfoLog()` have to be provided too.

6.2.4.3 Threading:

- `GTEST_HAS_NOTIFICATION_` - Enabled if Notification is already provided.
- `GTEST_HAS_MUTEX_AND_THREAD_LOCAL_` - Enabled if Mutex and ThreadLocal are already provided. Must also provide `GTEST_DECLARE_STATIC_MUTEX_(mutex)` and `GTEST_DEFINE_STATIC_MUTEX_(mutex)`.
- `GTEST_EXCLUSIVE_LOCK_REQUIRED_(locks)`
- `GTEST_LOCK_EXCLUDED_(locks)`

6.2.4.4 Underlying library support features

- `GTEST_HAS_CXXABI_H_`

6.2.4.5 Exporting API symbols:

- `GTEST_API_` - Specifier for exported symbols.

6.2.5 Header `gtest-printers.h`

- See documentation at `gtest/gtest-printers.h` for details on how to define a custom printer.

6.3 Dokumentacja katalogu packages/Microsoft.googletest.v140.[←](#) windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest

Katalogi

- katalog [internal](#)

Pliki

- plik [gtest-death-test.h](#)
- plik [gtest-message.h](#)
- plik [gtest-param-test.h](#)
- plik [gtest-printers.h](#)
- plik [gtest-spi.h](#)
- plik [gtest-test-part.h](#)
- plik [gtest-typed-test.h](#)
- plik [gtest.h](#)
- plik [gtest_pred_impl.h](#)
- plik [gtest_prod.h](#)

6.4 Dokumentacja katalogu packages/Microsoft.googletest.v140.[←](#) windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include

Katalogi

- katalog [gtest](#)

6.5 Dokumentacja katalogu packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt- dyn.1.8.1.8/build/native/include/gtest/internal

Katalogi

- katalog [custom](#)

Pliki

- plik [gtest-death-test-internal.h](#)
- plik [gtest-filepath.h](#)
- plik [gtest-internal.h](#)
- plik [gtest-linked_ptr.h](#)
- plik [gtest-param-util-generated.h](#)
- plik [gtest-param-util.h](#)
- plik [gtest-port-arch.h](#)
- plik [gtest-port.h](#)
- plik [gtest-string.h](#)
- plik [gtest-tuple.h](#)
- plik [gtest-type-util.h](#)

6.6 Dokumentacja katalogu MergeSortApp

Pliki

- plik [MergeSort.h](#)
Implementacja algorytmu sortowania przez scalanie (Merge Sort).
- plik [MergeSortApp.cpp](#)
Główny plik programu uruchamiający algorytm.

6.7 Dokumentacja katalogu packages/Microsoft.goolgeTest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8

Katalogi

- katalog [build](#)

6.8 Dokumentacja katalogu packages/Microsoft.goolgeTest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native

Katalogi

- katalog [include](#)

6.9 Dokumentacja katalogu packages

Katalogi

- katalog [Microsoft.goolgeTest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8](#)

Rozdział 7

Dokumentacja przestrzeni nazw

7.1 Dokumentacja przestrzeni nazw proto2

7.2 Dokumentacja przestrzeni nazw std

Przestrzenie nazw

- namespace [tr1](#)

7.3 Dokumentacja przestrzeni nazw std::tr1

Przestrzenie nazw

- namespace [gtest_internal](#)

Komponenty

- class [tuple](#)
- class [tuple<>](#)
- struct [tuple_size](#)
- struct [tuple_size< GTEST_0_TUPLE_\(T\) >](#)
- struct [tuple_size< GTEST_1_TUPLE_\(T\) >](#)
- struct [tuple_size< GTEST_2_TUPLE_\(T\) >](#)
- struct [tuple_size< GTEST_3_TUPLE_\(T\) >](#)
- struct [tuple_size< GTEST_4_TUPLE_\(T\) >](#)
- struct [tuple_size< GTEST_5_TUPLE_\(T\) >](#)
- struct [tuple_size< GTEST_6_TUPLE_\(T\) >](#)
- struct [tuple_size< GTEST_7_TUPLE_\(T\) >](#)
- struct [tuple_size< GTEST_8_TUPLE_\(T\) >](#)
- struct [tuple_size< GTEST_9_TUPLE_\(T\) >](#)
- struct [tuple_size< GTEST_10_TUPLE_\(T\) >](#)
- struct [tuple_element](#)

Funkcje

- template<GTEST_1_TYPENAMES_(T)>
class GTEST_1_TUPLE_(T)
- template<GTEST_2_TYPENAMES_(T)>
class GTEST_2_TUPLE_(T)
- template<GTEST_3_TYPENAMES_(T)>
class GTEST_3_TUPLE_(T)
- template<GTEST_4_TYPENAMES_(T)>
class GTEST_4_TUPLE_(T)
- template<GTEST_5_TYPENAMES_(T)>
class GTEST_5_TUPLE_(T)
- template<GTEST_6_TYPENAMES_(T)>
class GTEST_6_TUPLE_(T)
- template<GTEST_7_TYPENAMES_(T)>
class GTEST_7_TUPLE_(T)
- template<GTEST_8_TYPENAMES_(T)>
class GTEST_8_TUPLE_(T)
- template<GTEST_9_TYPENAMES_(T)>
class GTEST_9_TUPLE_(T)
- tuple make_tuple ()
- template<GTEST_1_TYPENAMES_(T)>
GTEST_1_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_2_TYPENAMES_(T)>
GTEST_2_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_3_TYPENAMES_(T)>
GTEST_3_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_4_TYPENAMES_(T)>
GTEST_4_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_5_TYPENAMES_(T)>
GTEST_5_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_6_TYPENAMES_(T)>
GTEST_6_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_7_TYPENAMES_(T)>
GTEST_7_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_8_TYPENAMES_(T)>
GTEST_8_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_9_TYPENAMES_(T)>
GTEST_9_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_10_TYPENAMES_(T)>
GTEST_10_TUPLE_(T) make_tuple(const T0 &f0)
- template<int k, GTEST_10_TYPENAMES_(T)>
GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(k, GTEST_10_TUPLE_(T))) get(GTEST_10_TUPLE_(T)
&t)
- template<int k, GTEST_10_TYPENAMES_(T)>
GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(k, GTEST_10_TUPLE_(T))) get(const GTEST_10_TUPLE_(T)
&t)
- template<GTEST_10_TYPENAMES_(T), GTEST_10_TYPENAMES_(U)>
bool operator==(const GTEST_10_TUPLE_(T)&t, const GTEST_10_TUPLE_(U)&u)
- template<GTEST_10_TYPENAMES_(T), GTEST_10_TYPENAMES_(U)>
bool operator!=(const GTEST_10_TUPLE_(T)&t, const GTEST_10_TUPLE_(U)&u)

Zmienne

- const T1 & **f1**
- const T1 const T2 & **f2**
- const T1 const T2 const T3 & **f3**
- const T1 const T2 const T3 const T4 & **f4**
- const T1 const T2 const T3 const T4 const T5 & **f5**
- const T1 const T2 const T3 const T4 const T5 const T6 & **f6**
- const T1 const T2 const T3 const T4 const T5 const T6 const T7 & **f7**
- const T1 const T2 const T3 const T4 const T5 const T6 const T7 const T8 & **f8**
- const T1 const T2 const T3 const T4 const T5 const T6 const T7 const T8 const T9 & **f9**

7.3.1 Dokumentacja funkcji

7.3.1.1 GTEST_10_TUPLE_()

```
template<GTEST_10_TYPENAMES_(T)>
std::tr1::GTEST_10_TUPLE_(
    T ) const & [inline]
```

7.3.1.2 GTEST_1_TUPLE_() [1/2]

```
template<GTEST_1_TYPENAMES_(T)>
class std::tr1::GTEST_1_TUPLE_(
    T )
```

7.3.1.3 GTEST_1_TUPLE_() [2/2]

```
template<GTEST_1_TYPENAMES_(T)>
std::tr1::GTEST_1_TUPLE_(
    T ) const & [inline]
```

7.3.1.4 GTEST_2_TUPLE_() [1/2]

```
template<GTEST_2_TYPENAMES_(T)>
class std::tr1::GTEST_2_TUPLE_(
    T )
```

7.3.1.5 GTEST_2_TUPLE_() [2/2]

```
template<GTEST_2_TYPENAMES_(T)>
std::tr1::GTEST_2_TUPLE_(
    T ) const & [inline]
```

7.3.1.6 GTEST_3_TUPLE_() [1/2]

```
template<GTEST_3_TYPENAMES_(T)>
class std::tr1::GTEST_3_TUPLE_
{
    T
}
```

7.3.1.7 GTEST_3_TUPLE_() [2/2]

```
template<GTEST_3_TYPENAMES_(T)>
std::tr1::GTEST_3_TUPLE_
{
    T const & [inline]
}
```

7.3.1.8 GTEST_4_TUPLE_() [1/2]

```
template<GTEST_4_TYPENAMES_(T)>
class std::tr1::GTEST_4_TUPLE_
{
    T
}
```

7.3.1.9 GTEST_4_TUPLE_() [2/2]

```
template<GTEST_4_TYPENAMES_(T)>
std::tr1::GTEST_4_TUPLE_
{
    T const & [inline]
}
```

7.3.1.10 GTEST_5_TUPLE_() [1/2]

```
template<GTEST_5_TYPENAMES_(T)>
class std::tr1::GTEST_5_TUPLE_
{
    T
}
```

7.3.1.11 GTEST_5_TUPLE_() [2/2]

```
template<GTEST_5_TYPENAMES_(T)>
std::tr1::GTEST_5_TUPLE_
{
    T const & [inline]
}
```

7.3.1.12 GTEST_6_TUPLE_() [1/2]

```
template<GTEST_6_TYPENAMES_(T)>
class std::tr1::GTEST_6_TUPLE_
{
    T
}
```

7.3.1.13 GTEST_6_TUPLE_() [2/2]

```
template<GTEST_6_TYPENAMES_(T)>
std::tr1::GTEST_6_TUPLE_
{
    T const & [inline]
}
```

7.3.1.14 GTEST_7_TUPLE_() [1/2]

```
template<GTEST_7_TYPENAMES_(T)>
class std::tr1::GTEST_7_TUPLE_
{
    T
}
```

7.3.1.15 GTEST_7_TUPLE_() [2/2]

```
template<GTEST_7_TYPENAMES_(T)>
std::tr1::GTEST_7_TUPLE_
{
    T const & [inline]
}
```

7.3.1.16 GTEST_8_TUPLE_() [1/2]

```
template<GTEST_8_TYPENAMES_(T)>
class std::tr1::GTEST_8_TUPLE_
{
    T
}
```

7.3.1.17 GTEST_8_TUPLE_() [2/2]

```
template<GTEST_8_TYPENAMES_(T)>
std::tr1::GTEST_8_TUPLE_
{
    T const & [inline]
}
```

7.3.1.18 GTEST_9_TUPLE_() [1/2]

```
template<GTEST_9_TYPENAMES_(T)>
class std::tr1::GTEST_9_TUPLE_
{
    T
}
```

7.3.1.19 GTEST_9_TUPLE_() [2/2]

```
template<GTEST_9_TYPENAMES_(T)>
std::tr1::GTEST_9_TUPLE_
{
    T const & [inline]
}
```

7.3.1.20 GTEST_ADD_REF_()

```
template<int k, GTEST_10_TYPENAMES_(T)>
std::tr1::GTEST_ADD_REF_
{
    GTEST_TUPLE_ELEMENT_(k, GTEST_10_TUPLE_(T)) &
}
```

7.3.1.21 GTEST_BY_REF_()

```
template<int k, GTEST_10_TYPENAMES_(T)>
std::tr1::GTEST_BY_REF_
{
    GTEST_TUPLE_ELEMENT_(k, GTEST_10_TUPLE_(T)) const &
```

7.3.1.22 make_tuple()

```
tuple std::tr1::make_tuple () [inline]
```

7.3.1.23 operator"!=()

```
template<GTEST_10_TYPENAMES_(T), GTEST_10_TYPENAMES_(U)>
bool std::tr1::operator!= (
    const GTEST_10_TUPLE_(T) & t,
    const GTEST_10_TUPLE_(U) & u) [inline]
```

7.3.1.24 operator==()

```
template<GTEST_10_TYPENAMES_(T), GTEST_10_TYPENAMES_(U)>
bool std::tr1::operator== (
    const GTEST_10_TUPLE_(T) & t,
    const GTEST_10_TUPLE_(U) & u) [inline]
```

7.3.2 Dokumentacja zmiennych

7.3.2.1 f1

```
const T1 & std::tr1::f1
```

Wartość początkowa:

```
{
    return GTEST_2_TUPLE_(T)(f0, f1)
```

7.3.2.2 f2

```
const T1 const T2 & std::tr1::f2
```

Wartość początkowa:

```
{
    return GTEST_3_TUPLE_(T)(f0, f1, f2)
```

7.3.2.3 f3

```
const T1 const T2 const T3 & std::tr1::f3
```

Wartość początkowa:

```
{
    return GTEST_4_TUPLE_(T)(f0, f1, f2, f3)
```

7.3.2.4 f4

```
const T1 const T2 const T3 const T4 & std::tr1::f4
```

Wartość początkowa:

```
{
    return GTEST_5_TUPLE_(T)(f0, f1, f2, f3, f4)
```

7.3.2.5 f5

```
const T1 const T2 const T3 const T4 const T5 & std::tr1::f5
```

Wartość początkowa:

```
{  
    return GTEST_6_TUPLE_(T)(f0, f1, f2, f3, f4, f5)
```

7.3.2.6 f6

```
const T1 const T2 const T3 const T4 const T5 const T6 & std::tr1::f6
```

Wartość początkowa:

```
{  
    return GTEST_7_TUPLE_(T)(f0, f1, f2, f3, f4, f5, f6)
```

7.3.2.7 f7

```
const T1 const T2 const T3 const T4 const T5 const T6 const T7 & std::tr1::f7
```

Wartość początkowa:

```
{  
    return GTEST_8_TUPLE_(T)(f0, f1, f2, f3, f4, f5, f6, f7)
```

7.3.2.8 f8

```
const T1 const T2 const T3 const T4 const T5 const T6 const T7 const T8 & std::tr1::f8
```

Wartość początkowa:

```
{  
    return GTEST_9_TUPLE_(T)(f0, f1, f2, f3, f4, f5, f6, f7, f8)
```

7.3.2.9 f9

```
const T1 const T2 const T3 const T4 const T5 const T6 const T7 const T8 const T9& std::tr1::f9
```

Wartość początkowa:

```
{  
    return GTEST_10_TUPLE_(T)(f0, f1, f2, f3, f4, f5, f6, f7, f8, f9)
```

7.4 Dokumentacja przestrzeni nazw std::tr1::gtest_internal

Komponenty

- struct [ByRef](#)
- struct [ByRef< T & >](#)
- struct [AddRef](#)
- struct [AddRef< T & >](#)
- class [Get](#)
- struct [TupleElement](#)
- struct [TupleElement< true, 0, GTEST_10_TUPLE_\(T\) >](#)
- struct [TupleElement< true, 1, GTEST_10_TUPLE_\(T\) >](#)
- struct [TupleElement< true, 2, GTEST_10_TUPLE_\(T\) >](#)
- struct [TupleElement< true, 3, GTEST_10_TUPLE_\(T\) >](#)
- struct [TupleElement< true, 4, GTEST_10_TUPLE_\(T\) >](#)
- struct [TupleElement< true, 5, GTEST_10_TUPLE_\(T\) >](#)
- struct [TupleElement< true, 6, GTEST_10_TUPLE_\(T\) >](#)
- struct [TupleElement< true, 7, GTEST_10_TUPLE_\(T\) >](#)
- struct [TupleElement< true, 8, GTEST_10_TUPLE_\(T\) >](#)
- struct [TupleElement< true, 9, GTEST_10_TUPLE_\(T\) >](#)
- class [Get< 0 >](#)
- class [Get< 1 >](#)
- class [Get< 2 >](#)
- class [Get< 3 >](#)
- class [Get< 4 >](#)
- class [Get< 5 >](#)
- class [Get< 6 >](#)
- class [Get< 7 >](#)
- class [Get< 8 >](#)
- class [Get< 9 >](#)
- struct [SameSizeTuplePrefixComparator](#)
- struct [SameSizeTuplePrefixComparator< 0, 0 >](#)
- struct [SameSizeTuplePrefixComparator< k, k >](#)

7.5 Dokumentacja przestrzeni nazw testing

Przestrzenie nazw

- namespace [internal](#)
- namespace [internal2](#)

Komponenty

- class [Message](#)
- class [Test](#)
- class [TestProperty](#)
- class [TestResult](#)
- class [TestInfo](#)
- class [TestCase](#)
- class [Environment](#)
- class [TestEventListener](#)

- class `EmptyTestEventListener`
- class `TestEventListeners`
- class `UnitTest`
- class `WithParamInterface`
- class `TestWithParam`
- class `ScopedTrace`
- struct `TestParamInfo`
- struct `PrintToStringParamName`

Definicje typów

- `typedef internal::TimeInMillis TimeInMillis`

Funkcje

- `GTEST_DECLARE_string_(death_test_style)`
- `std::ostream & operator<< (std::ostream &os, const Message &sb)`
- template<typename T, typename IncrementT>
`internal::ParamGenerator< T > Range (T start, T end, IncrementT step)`
- template<typename T>
`internal::ParamGenerator< T > Range (T start, T end)`
- template<typename ForwardIterator>
`internal::ParamGenerator< typename ::testing::internal::IteratorTraits< ForwardIterator >::value_type > ValuesIn (ForwardIterator begin, ForwardIterator end)`
- template<typename T, size_t N>
`internal::ParamGenerator< T > ValuesIn (const T(&array)[N])`
- template<class Container>
`internal::ParamGenerator< typename Container::value_type > ValuesIn (const Container &container)`
- template<typename T1>
`internal::ValueArray1< T1 > Values (T1 v1)`
- template<typename T1, typename T2>
`internal::ValueArray2< T1, T2 > Values (T1 v1, T2 v2)`
- template<typename T1, typename T2, typename T3>
`internal::ValueArray3< T1, T2, T3 > Values (T1 v1, T2 v2, T3 v3)`
- template<typename T1, typename T2, typename T3, typename T4>
`internal::ValueArray4< T1, T2, T3, T4 > Values (T1 v1, T2 v2, T3 v3, T4 v4)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5>
`internal::ValueArray5< T1, T2, T3, T4, T5 > Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>
`internal::ValueArray6< T1, T2, T3, T4, T5, T6 > Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7>
`internal::ValueArray7< T1, T2, T3, T4, T5, T6, T7 > Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8>
`internal::ValueArray8< T1, T2, T3, T4, T5, T6, T7, T8 > Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9>
`internal::ValueArray9< T1, T2, T3, T4, T5, T6, T7, T8, T9 > Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10>
`internal::ValueArray10< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 > Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10)`


```
internal::ValueArray22< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22 > Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22)
```

typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30>

internal::ValueArray30< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30 > **Values** (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30)

```
internal::ValueArray36< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36 > Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36)
```

v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41)

v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46)

- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44, typename T45, typename T46, typename T47>
`internal::ValueArray47< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47 > Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48>
`internal::ValueArray48< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48 > Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47, T48 v48)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48, typename T49>
`internal::ValueArray49< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49 > Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47, T48 v48, T49 v49)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48, typename T49, typename T50>
`internal::ValueArray50< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49, T50 > Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47, T48 v48, T49 v49, T50 v50)`
- `internal::ParamGenerator< bool > Bool ()`
- template<typename T>
`::std::string PrintToString (const T &value)`

- `Environment * AddGlobalTestEnvironment (Environment *env)`
- `GTEST_API_ void InitGoogleTest (int *argc, char **argv)`
- `GTEST_API_ void InitGoogleTest (int *argc, wchar_t **argv)`
- `GTEST_API_ AssertionResult IsSubstring (const char *needle_expr, const char *haystack_expr, const char *needle, const char *haystack)`
- `GTEST_API_ AssertionResult IsSubstring (const char *needle_expr, const char *haystack_expr, const wchar_t *needle, const wchar_t *haystack)`
- `GTEST_API_ AssertionResult IsNotSubstring (const char *needle_expr, const char *haystack_expr, const char *needle, const char *haystack)`
- `GTEST_API_ AssertionResult IsNotSubstring (const char *needle_expr, const char *haystack_expr, const wchar_t *needle, const wchar_t *haystack)`
- `GTEST_API_ AssertionResult IsSubstring (const char *needle_expr, const char *haystack_expr, const ::std::string &needle, const ::std::string &haystack)`
- `GTEST_API_ AssertionResult IsNotSubstring (const char *needle_expr, const char *haystack_expr, const ::std::string &needle, const ::std::string &haystack)`
- `GTEST_API_ AssertionResult FloatLE (const char *expr1, const char *expr2, float val1, float val2)`
- `GTEST_API_ AssertionResult DoubleLE (const char *expr1, const char *expr2, double val1, double val2)`
- template<typename T1, typename T2>
 `bool StaticAssertTypeEq ()`
- `GTEST_API_ std::string TempDir ()`
- template<typename Pred, typename T1>
 `AssertionResult AssertPred1Helper (const char *pred_text, const char *e1, Pred pred, const T1 &v1)`
- template<typename Pred, typename T1, typename T2>
 `AssertionResult AssertPred2Helper (const char *pred_text, const char *e1, const char *e2, Pred pred, const T1 &v1, const T2 &v2)`
- template<typename Pred, typename T1, typename T2, typename T3>
 `AssertionResult AssertPred3Helper (const char *pred_text, const char *e1, const char *e2, const char *e3, Pred pred, const T1 &v1, const T2 &v2, const T3 &v3)`
- template<typename Pred, typename T1, typename T2, typename T3, typename T4>
 `AssertionResult AssertPred4Helper (const char *pred_text, const char *e1, const char *e2, const char *e3, const char *e4, Pred pred, const T1 &v1, const T2 &v2, const T3 &v3, const T4 &v4)`
- template<typename Pred, typename T1, typename T2, typename T3, typename T4, typename T5>
 `AssertionResult AssertPred5Helper (const char *pred_text, const char *e1, const char *e2, const char *e3, const char *e4, const char *e5, Pred pred, const T1 &v1, const T2 &v2, const T3 &v3, const T4 &v4, const T5 &v5)`

Zmienne

- template<typename T>
 `const T * WithParamInterface< T >::parameter_ = NULL`
- class `GTEST_API_ testing::ScopedTrace GTEST_ATTRIBUTE_UNUSED_`

7.5.1 Dokumentacja definicji typów

7.5.1.1 TimeInMillis

```
typedef internal::TimeInMillis testing::TimeInMillis
```

7.5.2 Dokumentacja funkcji

7.5.2.1 AddGlobalTestEnvironment()

```
Environment * testing::AddGlobalTestEnvironment (
    Environment * env) [inline]
```

7.5.2.2 AssertPred1Helper()

```
template<typename Pred, typename T1>
AssertionResult testing::AssertPred1Helper (
    const char * pred_text,
    const char * e1,
    Pred pred,
    const T1 & v1)
```

7.5.2.3 AssertPred2Helper()

```
template<typename Pred, typename T1, typename T2>
AssertionResult testing::AssertPred2Helper (
    const char * pred_text,
    const char * e1,
    const char * e2,
    Pred pred,
    const T1 & v1,
    const T2 & v2)
```

7.5.2.4 AssertPred3Helper()

```
template<typename Pred, typename T1, typename T2, typename T3>
AssertionResult testing::AssertPred3Helper (
    const char * pred_text,
    const char * e1,
    const char * e2,
    const char * e3,
    Pred pred,
    const T1 & v1,
    const T2 & v2,
    const T3 & v3)
```

7.5.2.5 AssertPred4Helper()

```
template<typename Pred, typename T1, typename T2, typename T3, typename T4>
AssertionResult testing::AssertPred4Helper (
    const char * pred_text,
    const char * e1,
    const char * e2,
    const char * e3,
    const char * e4,
    Pred pred,
    const T1 & v1,
    const T2 & v2,
    const T3 & v3,
    const T4 & v4)
```

7.5.2.6 AssertPred5Helper()

```
template<typename Pred, typename T1, typename T2, typename T3, typename T4, typename T5>
AssertionResult testing::AssertPred5Helper (
    const char * pred_text,
    const char * e1,
    const char * e2,
    const char * e3,
    const char * e4,
    const char * e5,
    Pred pred,
    const T1 & v1,
    const T2 & v2,
    const T3 & v3,
    const T4 & v4,
    const T5 & v5)
```

7.5.2.7 Bool()

```
internal::ParamGenerator< bool > testing::Bool () [inline]
```

7.5.2.8 DoubleLE()

```
GTEST_API_ AssertionResult testing::DoubleLE (
    const char * expr1,
    const char * expr2,
    double val1,
    double val2)
```

7.5.2.9 FloatLE()

```
GTEST_API_ AssertionResult testing::FloatLE (
    const char * expr1,
    const char * expr2,
    float val1,
    float val2)
```

7.5.2.10 GTEST_DECLARE_string_()

```
testing::GTEST_DECLARE_string_ (
    death_test_style )
```

7.5.2.11 InitGoogleTest() [1/2]

```
GTEST_API_ void testing::InitGoogleTest (
    int * argc,
    char ** argv)
```

7.5.2.12 InitGoogleTest() [2/2]

```
GTEST_API_ void testing::InitGoogleTest (
    int * argc,
    wchar_t ** argv)
```

7.5.2.13 IsNotSubstring() [1/3]

```
GTEST_API_ AssertionResult testing::IsNotSubstring (
    const char * needle_expr,
    const char * haystack_expr,
    const ::std::string & needle,
    const ::std::string & haystack)
```

7.5.2.14 IsNotSubstring() [2/3]

```
GTEST_API_ AssertionResult testing::IsNotSubstring (
    const char * needle_expr,
    const char * haystack_expr,
    const char * needle,
    const char * haystack)
```

7.5.2.15 IsNotSubstring() [3/3]

```
GTEST_API_ AssertionResult testing::IsNotSubstring (
    const char * needle_expr,
    const char * haystack_expr,
    const wchar_t * needle,
    const wchar_t * haystack)
```

7.5.2.16 IsSubstring() [1/3]

```
GTEST_API_ AssertionResult testing::IsSubstring (
    const char * needle_expr,
    const char * haystack_expr,
    const ::std::string & needle,
    const ::std::string & haystack)
```

7.5.2.17 IsSubstring() [2/3]

```
GTEST_API_ AssertionResult testing::IsSubstring (
    const char * needle_expr,
    const char * haystack_expr,
    const char * needle,
    const char * haystack)
```

7.5.2.18 **IsSubstring()** [3/3]

```
GTEST_API_ AssertionResult testing::IsSubstring (
    const char * needle_expr,
    const char * haystack_expr,
    const wchar_t * needle,
    const wchar_t * haystack)
```

7.5.2.19 **operator<<()**

```
std::ostream & testing::operator<< (
    std::ostream & os,
    const Message & sb) [inline]
```

7.5.2.20 **PrintToString()**

```
template<typename T>
::std::string testing::PrintToString (
    const T & value)
```

7.5.2.21 **Range()** [1/2]

```
template<typename T>
internal::ParamGenerator< T > testing::Range (
    T start,
    T end)
```

7.5.2.22 **Range()** [2/2]

```
template<typename T, typename IncrementT>
internal::ParamGenerator< T > testing::Range (
    T start,
    T end,
    IncrementT step)
```

7.5.2.23 **StaticAssertTypeEq()**

```
template<typename T1, typename T2>
bool testing::StaticAssertTypeEq ()
```

7.5.2.24 **TempDir()**

```
GTEST_API_ std::string testing::TempDir ()
```

7.5.2.25 Values() [1/50]

```
template<typename T1>
internal::ValueArray1< T1 > testing::Values (
    T1 v1)
```

7.5.2.26 Values() [2/50]

```
template<typename T1, typename T2>
internal::ValueArray2< T1, T2 > testing::Values (
    T1 v1,
    T2 v2)
```

7.5.2.27 Values() [3/50]

```
template<typename T1, typename T2, typename T3>
internal::ValueArray3< T1, T2, T3 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3)
```

7.5.2.28 Values() [4/50]

```
template<typename T1, typename T2, typename T3, typename T4>
internal::ValueArray4< T1, T2, T3, T4 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4)
```

7.5.2.29 Values() [5/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5>
internal::ValueArray5< T1, T2, T3, T4, T5 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5)
```

7.5.2.30 Values() [6/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>
internal::ValueArray6< T1, T2, T3, T4, T5, T6 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6)
```

7.5.2.31 Values() [7/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7>
internal::ValueArray7< T1, T2, T3, T4, T5, T6, T7 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7)
```

7.5.2.32 Values() [8/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8>
internal::ValueArray8< T1, T2, T3, T4, T5, T6, T7, T8 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8)
```

7.5.2.33 Values() [9/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9>
internal::ValueArray9< T1, T2, T3, T4, T5, T6, T7, T8, T9 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9)
```

7.5.2.34 Values() [10/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10>
internal::ValueArray10< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10)
```

7.5.2.35 Values() [11/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11>
internal::ValueArray11< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11)
```

7.5.2.36 Values() [12/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12>
internal::ValueArray12< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12)
```

7.5.2.37 Values() [13/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13>
internal::ValueArray13< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13)
```

7.5.2.38 Values() [14/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14>
internal::ValueArray14< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14 > testing::
::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14)
```

7.5.2.39 Values() [15/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15>
internal::ValueArray15< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15 >
testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15)
```

7.5.2.40 Values() [16/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16>
internal::ValueArray16< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16
> testing::Values (
    T1 v1,
    T2 v2,
```

```
T3 v3,
T4 v4,
T5 v5,
T6 v6,
T7 v7,
T8 v8,
T9 v9,
T10 v10,
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16)
```

7.5.2.41 Values() [17/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17>
internal::ValueArray17< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17)
```

7.5.2.42 Values() [18/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18>
internal::ValueArray18< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
```

```
T9 v9,
T10 v10,
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18)
```

7.5.2.43 Values() [19/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19>
internal::ValueArray19< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19)
```

7.5.2.44 Values() [20/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20>
internal::ValueArray20< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
```

```
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20)
```

7.5.2.45 Values() [21/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21>
internal::ValueArray21< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21)
```

7.5.2.46 Values() [22/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22>
internal::ValueArray22< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
```

```

T7 v7,
T8 v8,
T9 v9,
T10 v10,
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22)

```

7.5.2.47 Values() [23/50]

```

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23>
internal::ValueArray23< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23)

```

7.5.2.48 Values() [24/50]

```

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24>

```

```
internal::ValueArray24< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24)
```

7.5.2.49 Values() [25/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25> internal::ValueArray25< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24, T25 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
```

```
T23 v23,  
T24 v24,  
T25 v25)
```

7.5.2.50 Values() [26/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26>  
internal::ValueArray26< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,  
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26 > testing::Values (
```

T1 v1,
T2 v2,
T3 v3,
T4 v4,
T5 v5,
T6 v6,
T7 v7,
T8 v8,
T9 v9,
T10 v10,
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26)

7.5.2.51 Values() [27/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27>  
internal::ValueArray27< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,  
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27 > testing::Values (
```

T1 v1,
T2 v2,
T3 v3,
T4 v4,
T5 v5,
T6 v6,
T7 v7,
T8 v8,

```
T9 v9,
T10 v10,
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27)
```

7.5.2.52 Values() [28/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28>
internal::ValueArray28< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27,
    T28 v28)
```

7.5.2.53 Values() [29/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29>
internal::ValueArray29< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27,
    T28 v28,
    T29 v29)
```

7.5.2.54 Values() [30/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30>
internal::ValueArray30< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
```

```
T9 v9,
T10 v10,
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30)
```

7.5.2.55 Values() [31/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
         typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14,
         typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21,
         typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28,
         typename T29, typename T30, typename T31>
internal::ValueArray31< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
```

```
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30,
T31 v31)
```

7.5.2.56 Values() [32/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32>
internal::ValueArray32< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32 > testing::←
Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27,
    T28 v28,
    T29 v29,
    T30 v30,
    T31 v31,
    T32 v32)
```

7.5.2.57 Values() [33/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
```

```
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33>  
internal::ValueArray33< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,  
T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33 >  
testing::Values (   
    T1 v1,  
    T2 v2,  
    T3 v3,  
    T4 v4,  
    T5 v5,  
    T6 v6,  
    T7 v7,  
    T8 v8,  
    T9 v9,  
    T10 v10,  
    T11 v11,  
    T12 v12,  
    T13 v13,  
    T14 v14,  
    T15 v15,  
    T16 v16,  
    T17 v17,  
    T18 v18,  
    T19 v19,  
    T20 v20,  
    T21 v21,  
    T22 v22,  
    T23 v23,  
    T24 v24,  
    T25 v25,  
    T26 v26,  
    T27 v27,  
    T28 v28,  
    T29 v29,  
    T30 v30,  
    T31 v31,  
    T32 v32,  
    T33 v33)
```

7.5.2.58 Values() [34/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34>  
internal::ValueArray34< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,  
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34 >  
testing::Values (   
    T1 v1,  
    T2 v2,  
    T3 v3,  
    T4 v4,  
    T5 v5,  
    T6 v6,  
    T7 v7,  
    T8 v8,
```

```

T9 v9,
T10 v10,
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30,
T31 v31,
T32 v32,
T33 v33,
T34 v34)

```

7.5.2.59 Values() [35/50]

```

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35>
internal::ValueArray35< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35
> testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,

```

```
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30,
T31 v31,
T32 v32,
T33 v33,
T34 v34,
T35 v35)
```

7.5.2.60 Values() [36/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36>
internal::ValueArray36< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,
T36 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27,
    T28 v28,
    T29 v29,
```

```
T30 v30,
T31 v31,
T32 v32,
T33 v33,
T34 v34,
T35 v35,
T36 v36)
```

7.5.2.61 Values() [37/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37>
internal::ValueArray<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,
T36, T37> testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27,
    T28 v28,
    T29 v29,
    T30 v30,
    T31 v31,
    T32 v32,
    T33 v33,
    T34 v34,
    T35 v35,
    T36 v36,
    T37 v37)
```

7.5.2.62 Values() [38/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35, typename T36, typename T37, typename T38> testing::Values()  
internal::ValueArray38< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,  
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,  
T36, T37, T38 > testing::Values()  
    T1 v1,  
    T2 v2,  
    T3 v3,  
    T4 v4,  
    T5 v5,  
    T6 v6,  
    T7 v7,  
    T8 v8,  
    T9 v9,  
    T10 v10,  
    T11 v11,  
    T12 v12,  
    T13 v13,  
    T14 v14,  
    T15 v15,  
    T16 v16,  
    T17 v17,  
    T18 v18,  
    T19 v19,  
    T20 v20,  
    T21 v21,  
    T22 v22,  
    T23 v23,  
    T24 v24,  
    T25 v25,  
    T26 v26,  
    T27 v27,  
    T28 v28,  
    T29 v29,  
    T30 v30,  
    T31 v31,  
    T32 v32,  
    T33 v33,  
    T34 v34,  
    T35 v35,  
    T36 v36,  
    T37 v37,  
    T38 v38)
```

7.5.2.63 Values() [39/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35, typename T36, typename T37, typename T38> testing::Values()  
internal::ValueArray38< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,  
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,  
T36, T37, T38 > testing::Values()
```

```
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39>
internal::ValueArray39< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,
T36, T37, T38, T39 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27,
    T28 v28,
    T29 v29,
    T30 v30,
    T31 v31,
    T32 v32,
    T33 v33,
    T34 v34,
    T35 v35,
    T36 v36,
    T37 v37,
    T38 v38,
    T39 v39)
```

7.5.2.64 Values() [40/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40>
internal::ValueArray40< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,
T36, T37, T38, T39, T40 > testing::Values (
    T1 v1,
```

```
T2 v2,
T3 v3,
T4 v4,
T5 v5,
T6 v6,
T7 v7,
T8 v8,
T9 v9,
T10 v10,
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30,
T31 v31,
T32 v32,
T33 v33,
T34 v34,
T35 v35,
T36 v36,
T37 v37,
T38 v38,
T39 v39,
T40 v40)
```

7.5.2.65 Values() [41/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41>
internal::ValueArray41< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,
T36, T37, T38, T39, T40, T41 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
```

```
T7 v7,  
T8 v8,  
T9 v9,  
T10 v10,  
T11 v11,  
T12 v12,  
T13 v13,  
T14 v14,  
T15 v15,  
T16 v16,  
T17 v17,  
T18 v18,  
T19 v19,  
T20 v20,  
T21 v21,  
T22 v22,  
T23 v23,  
T24 v24,  
T25 v25,  
T26 v26,  
T27 v27,  
T28 v28,  
T29 v29,  
T30 v30,  
T31 v31,  
T32 v32,  
T33 v33,  
T34 v34,  
T35 v35,  
T36 v36,  
T37 v37,  
T38 v38,  
T39 v39,  
T40 v40,  
T41 v41)
```

7.5.2.66 Values() [42/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename  
T42>  
internal::ValueArray42< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,  
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,  
T36, T37, T38, T39, T40, T41, T42 > testing::Values (  
    T1 v1,  
    T2 v2,  
    T3 v3,  
    T4 v4,  
    T5 v5,  
    T6 v6,  
    T7 v7,  
    T8 v8,  
    T9 v9,
```

```
T10 v10,  
T11 v11,  
T12 v12,  
T13 v13,  
T14 v14,  
T15 v15,  
T16 v16,  
T17 v17,  
T18 v18,  
T19 v19,  
T20 v20,  
T21 v21,  
T22 v22,  
T23 v23,  
T24 v24,  
T25 v25,  
T26 v26,  
T27 v27,  
T28 v28,  
T29 v29,  
T30 v30,  
T31 v31,  
T32 v32,  
T33 v33,  
T34 v34,  
T35 v35,  
T36 v36,  
T37 v37,  
T38 v38,  
T39 v39,  
T40 v40,  
T41 v41,  
T42 v42)
```

7.5.2.67 Values() [43/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename  
T42, typename T43>  
internal::ValueArray43< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,  
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,  
T36, T37, T38, T39, T40, T41, T42, T43 > testing::Values (
```

```
    T1 v1,  
    T2 v2,  
    T3 v3,  
    T4 v4,  
    T5 v5,  
    T6 v6,  
    T7 v7,  
    T8 v8,  
    T9 v9,  
    T10 v10,  
    T11 v11,
```

```
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30,
T31 v31,
T32 v32,
T33 v33,
T34 v34,
T35 v35,
T36 v36,
T37 v37,
T38 v38,
T39 v39,
T40 v40,
T41 v41,
T42 v42,
T43 v43)
```

7.5.2.68 Values() [44/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename
T42, typename T43, typename T44>
internal::ValueArray44< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,
T36, T37, T38, T39, T40, T41, T42, T43, T44 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
```

```
T13 v13,  
T14 v14,  
T15 v15,  
T16 v16,  
T17 v17,  
T18 v18,  
T19 v19,  
T20 v20,  
T21 v21,  
T22 v22,  
T23 v23,  
T24 v24,  
T25 v25,  
T26 v26,  
T27 v27,  
T28 v28,  
T29 v29,  
T30 v30,  
T31 v31,  
T32 v32,  
T33 v33,  
T34 v34,  
T35 v35,  
T36 v36,  
T37 v37,  
T38 v38,  
T39 v39,  
T40 v40,  
T41 v41,  
T42 v42,  
T43 v43,  
T44 v44)
```

7.5.2.69 Values() [45/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename  
T42, typename T43, typename T44, typename T45>  
internal::ValueArray45< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,  
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,  
T36, T37, T38, T39, T40, T41, T42, T43, T44, T45 > testing::Values (
```

T1 v1,
T2 v2,
T3 v3,
T4 v4,
T5 v5,
T6 v6,
T7 v7,
T8 v8,
T9 v9,
T10 v10,
T11 v11,
T12 v12,

```

T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30,
T31 v31,
T32 v32,
T33 v33,
T34 v34,
T35 v35,
T36 v36,
T37 v37,
T38 v38,
T39 v39,
T40 v40,
T41 v41,
T42 v42,
T43 v43,
T44 v44,
T45 v45)

```

7.5.2.70 Values() [46/50]

```

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename
T42, typename T43, typename T44, typename T45, typename T46>
internal::ValueArray46< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,
T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,

```

```
T12 v12,  
T13 v13,  
T14 v14,  
T15 v15,  
T16 v16,  
T17 v17,  
T18 v18,  
T19 v19,  
T20 v20,  
T21 v21,  
T22 v22,  
T23 v23,  
T24 v24,  
T25 v25,  
T26 v26,  
T27 v27,  
T28 v28,  
T29 v29,  
T30 v30,  
T31 v31,  
T32 v32,  
T33 v33,  
T34 v34,  
T35 v35,  
T36 v36,  
T37 v37,  
T38 v38,  
T39 v39,  
T40 v40,  
T41 v41,  
T42 v42,  
T43 v43,  
T44 v44,  
T45 v45,  
T46 v46)
```

7.5.2.71 Values() [47/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename  
T42, typename T43, typename T44, typename T45, typename T46, typename T47>  
internal::ValueArray47< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,  
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,  
T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47 > testing::Values (
```

T1 v1,
T2 v2,
T3 v3,
T4 v4,
T5 v5,
T6 v6,
T7 v7,
T8 v8,
T9 v9,

```
T10 v10,
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30,
T31 v31,
T32 v32,
T33 v33,
T34 v34,
T35 v35,
T36 v36,
T37 v37,
T38 v38,
T39 v39,
T40 v40,
T41 v41,
T42 v42,
T43 v43,
T44 v44,
T45 v45,
T46 v46,
T47 v47)
```

7.5.2.72 Values() [48/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename
T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48>
internal::ValueArray48< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,
T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48 > testing::Values (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
```

```
T7 v7,  
T8 v8,  
T9 v9,  
T10 v10,  
T11 v11,  
T12 v12,  
T13 v13,  
T14 v14,  
T15 v15,  
T16 v16,  
T17 v17,  
T18 v18,  
T19 v19,  
T20 v20,  
T21 v21,  
T22 v22,  
T23 v23,  
T24 v24,  
T25 v25,  
T26 v26,  
T27 v27,  
T28 v28,  
T29 v29,  
T30 v30,  
T31 v31,  
T32 v32,  
T33 v33,  
T34 v34,  
T35 v35,  
T36 v36,  
T37 v37,  
T38 v38,  
T39 v39,  
T40 v40,  
T41 v41,  
T42 v42,  
T43 v43,  
T44 v44,  
T45 v45,  
T46 v46,  
T47 v47,  
T48 v48)
```

7.5.2.73 Values() [49/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename  
T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48, typename  
T49>  
internal::ValueArray49< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,  
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,  
T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49 > testing::Values (
```

T1 v1,

```
T2 v2,
T3 v3,
T4 v4,
T5 v5,
T6 v6,
T7 v7,
T8 v8,
T9 v9,
T10 v10,
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30,
T31 v31,
T32 v32,
T33 v33,
T34 v34,
T35 v35,
T36 v36,
T37 v37,
T38 v38,
T39 v39,
T40 v40,
T41 v41,
T42 v42,
T43 v43,
T44 v44,
T45 v45,
T46 v46,
T47 v47,
T48 v48,
T49 v49)
```

7.5.2.74 Values() [50/50]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename
```

```
T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48, typename  
T49, typename T50>  
internal::ValueArray50< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16,  
T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35,  
T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49, T50 > testing::Values (   
    T1 v1,  
    T2 v2,  
    T3 v3,  
    T4 v4,  
    T5 v5,  
    T6 v6,  
    T7 v7,  
    T8 v8,  
    T9 v9,  
    T10 v10,  
    T11 v11,  
    T12 v12,  
    T13 v13,  
    T14 v14,  
    T15 v15,  
    T16 v16,  
    T17 v17,  
    T18 v18,  
    T19 v19,  
    T20 v20,  
    T21 v21,  
    T22 v22,  
    T23 v23,  
    T24 v24,  
    T25 v25,  
    T26 v26,  
    T27 v27,  
    T28 v28,  
    T29 v29,  
    T30 v30,  
    T31 v31,  
    T32 v32,  
    T33 v33,  
    T34 v34,  
    T35 v35,  
    T36 v36,  
    T37 v37,  
    T38 v38,  
    T39 v39,  
    T40 v40,  
    T41 v41,  
    T42 v42,  
    T43 v43,  
    T44 v44,  
    T45 v45,  
    T46 v46,  
    T47 v47,  
    T48 v48,  
    T49 v49,  
    T50 v50)
```

7.5.2.75 ValuesIn() [1/3]

```
template<class Container>
internal::ParamGenerator< typename Container::value_type > testing::ValuesIn (
    const Container & container)
```

7.5.2.76 ValuesIn() [2/3]

```
template<typename T, size_t N>
internal::ParamGenerator< T > testing::ValuesIn (
    const T(&) array[N])
```

7.5.2.77 ValuesIn() [3/3]

```
template<typename ForwardIterator>
internal::ParamGenerator< typename ::testing::internal::IteratorTraits< ForwardIterator >::value_type > testing::ValuesIn (
    ForwardIterator begin,
    ForwardIterator end)
```

7.5.3 Dokumentacja zmiennych

7.5.3.1 GTEST_ATTRIBUTE_UNUSED_

```
class GTEST_API_ testing::ScopedTrace testing::GTEST_ATTRIBUTE_UNUSED_
```

7.5.3.2 WithParamInterface< T >::parameter_

```
template<typename T>
const T* testing::WithParamInterface< T >::parameter_ = NULL
```

7.6 Dokumentacja przestrzeni nazw testing::internal

Przestrzenie nazw

- namespace `edit_distance`
- namespace `posix`

Komponenty

- class FormatForComparison
- class FormatForComparison< ToPrint[N], OtherOperand >
- class UniversalPrinter
- struct WrapPrinterType
- class UniversalPrinter< T[N]>
- class UniversalPrinter< T & >
- class UniversalTersePrinter
- class UniversalTersePrinter< T & >
- class UniversalTersePrinter< T[N]>
- class UniversalTersePrinter< const char * >
- class UniversalTersePrinter< char * >
- class UniversalTersePrinter< wchar_t * >
- struct TuplePolicy
- class EqHelper
- class EqHelper< true >
- class AssertHelper
- class FloatingPoint
- class TypeIdHelper
- class TestFactoryBase
- class TestFactoryImpl
- struct CodeLocation
- struct ConstCharPtr
- class Random
- struct CompileAssertTypesEqual
- struct CompileAssertTypesEqual< T, T >
- struct RemoveReference
- struct RemoveReference< T & >
- struct RemoveConst
- struct RemoveConst< const T >
- struct RemoveConst< const T[N]>
- class ImplicitlyConvertible
- struct IsAProtocolMessage
- struct IsHashTable
- struct VoidT
- struct HasValueType
- struct HasValueType< T, VoidT< typename T::value_type > >
- struct IsRecursiveContainerImpl
- struct IsRecursiveContainerImpl< C, false, HV >
- struct IsRecursiveContainerImpl< C, true, false >
- struct IsRecursiveContainerImpl< C, true, true >
- struct IsRecursiveContainer
- struct EnableIf
- struct EnableIf< true >
- struct RelationToSourceReference
- struct RelationToSourceCopy
- class NativeArray
- class linked_ptr_internal
- class linked_ptr
- class ValueArray1
- class ValueArray2
- class ValueArray3
- class ValueArray4
- class ValueArray5

- class [ValueArray6](#)
- class [ValueArray7](#)
- class [ValueArray8](#)
- class [ValueArray9](#)
- class [ValueArray10](#)
- class [ValueArray11](#)
- class [ValueArray12](#)
- class [ValueArray13](#)
- class [ValueArray14](#)
- class [ValueArray15](#)
- class [ValueArray16](#)
- class [ValueArray17](#)
- class [ValueArray18](#)
- class [ValueArray19](#)
- class [ValueArray20](#)
- class [ValueArray21](#)
- class [ValueArray22](#)
- class [ValueArray23](#)
- class [ValueArray24](#)
- class [ValueArray25](#)
- class [ValueArray26](#)
- class [ValueArray27](#)
- class [ValueArray28](#)
- class [ValueArray29](#)
- class [ValueArray30](#)
- class [ValueArray31](#)
- class [ValueArray32](#)
- class [ValueArray33](#)
- class [ValueArray34](#)
- class [ValueArray35](#)
- class [ValueArray36](#)
- class [ValueArray37](#)
- class [ValueArray38](#)
- class [ValueArray39](#)
- class [ValueArray40](#)
- class [ValueArray41](#)
- class [ValueArray42](#)
- class [ValueArray43](#)
- class [ValueArray44](#)
- class [ValueArray45](#)
- class [ValueArray46](#)
- class [ValueArray47](#)
- class [ValueArray48](#)
- class [ValueArray49](#)
- class [ValueArray50](#)
- class [ParamGeneratorInterface](#)
- class [ParamGenerator](#)
- class [ParamIteratorInterface](#)
- class [ParamIterator](#)
- class [RangeGenerator](#)
- class [ValuesInIteratorRangeGenerator](#)
- struct [ParamNameGenFunc](#)
- class [ParameterizedTestFactory](#)
- class [TestMetaFactoryBase](#)
- class [TestMetaFactory](#)

- class ParameterizedTestCaseInfoBase
- class ParameterizedTestCaseInfo
- class ParameterizedTestCaseRegistry
- struct CompileAssert
- struct StaticAssertTypeEqHelper
- struct StaticAssertTypeEqHelper< T, T >
- struct IsSame
- struct IsSame< T, T >
- class scoped_ptr
- class RE
- class GTestLog
- struct AddReference
- struct AddReference< T & >
- struct ConstRef
- struct ConstRef< T & >
- struct RvalueRef
- class Mutex
- class GTestMutexLock
- class ThreadLocal
- struct bool_constant
- struct is_same
- struct is_same< T, T >
- struct is_pointer
- struct is_pointer< T * >
- struct IteratorTraits
- struct IteratorTraits< T * >
- struct IteratorTraits< const T * >
- class TypeWithSize
- class TypeWithSize< 4 >
- class TypeWithSize< 8 >
- class String

Definicje typów

- typedef ::std::vector< ::std::string > Strings
- typedef FloatingPoint< float > Float
- typedef FloatingPoint< double > Double
- typedef const void * Typeld
- typedef void(* SetUpTestCaseFunc) ()
- typedef void(* TearDownTestCaseFunc) ()
- typedef int IsContainer
- typedef char IsNotContainer
- typedef ::std::string string
- typedef ::std::wstring wstring
- typedef GTestMutexLock MutexLock
- typedef bool_constant< false > false_type
- typedef bool_constant< true > true_type
- typedef long long BiggestInt
- typedef TypeWithSize< 4 >::Int Int32
- typedef TypeWithSize< 4 >::UInt UInt32
- typedef TypeWithSize< 8 >::Int Int64
- typedef TypeWithSize< 8 >::UInt UInt64
- typedef TypeWithSize< 8 >::Int TimeInMillis

Waliczenia

- enum `DefaultPrinterType` { `kPrintContainer` , `kPrintPointer` , `kPrintFunctionPointer` , `kPrintOther` }
- enum `GTestLogSeverity` { `GTEST_INFO` , `GTEST_WARNING` , `GTEST_ERROR` , `GTEST_FATAL` }

Funkcje

- template<typename T>
 `std::string StreamableToString` (const T &streamable)
- `GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_` (char)
- `GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_` (wchar_t)
- `GTEST_IMPL_FORMAT_C_STRING_AS_STRING_` (char, ::std::string)
- template<typename T1, typename T2>
 `std::string FormatForComparisonFailureMessage` (const T1 &`value`, const T2 &)
- template<typename T>
 void `UniversalPrint` (const T &`value`, ::std::ostream *os)
- template<typename C>
 void `DefaultPrintTo` (`WrapPrinterType<kPrintContainer>`, const C &container, ::std::ostream *os)
- template<typename T>
 void `DefaultPrintTo` (`WrapPrinterType<kPrintPointer>`, T *p, ::std::ostream *os)
- template<typename T>
 void `DefaultPrintTo` (`WrapPrinterType<kPrintFunctionPointer>`, T *p, ::std::ostream *os)
- template<typename T>
 void `DefaultPrintTo` (`WrapPrinterType<kPrintOther>`, const T &`value`, ::std::ostream *os)
- template<typename T>
 void `PrintTo` (const T &`value`, ::std::ostream *os)
- `GTEST_API_ void PrintTo` (unsigned char c, ::std::ostream *os)
- `GTEST_API_ void PrintTo` (signed char c, ::std::ostream *os)
- void `PrintTo` (char c, ::std::ostream *os)
- void `PrintTo` (bool x, ::std::ostream *os)
- `GTEST_API_ void PrintTo` (wchar_t wc, ::std::ostream *os)
- `GTEST_API_ void PrintTo` (const char *s, ::std::ostream *os)
- void `PrintTo` (char *s, ::std::ostream *os)
- void `PrintTo` (const signed char *s, ::std::ostream *os)
- void `PrintTo` (signed char *s, ::std::ostream *os)
- void `PrintTo` (const unsigned char *s, ::std::ostream *os)
- void `PrintTo` (unsigned char *s, ::std::ostream *os)
- `GTEST_API_ void PrintTo` (const wchar_t *s, ::std::ostream *os)
- void `PrintTo` (wchar_t *s, ::std::ostream *os)
- template<typename T>
 void `PrintRawArrayTo` (const T a[], size_t count, ::std::ostream *os)
- `GTEST_API_ void PrintStringTo` (const ::std::string &s, ::std::ostream *os)
- void `PrintTo` (const ::std::string &s, ::std::ostream *os)
- template<typename T1, typename T2>
 void `PrintTo` (const ::std::pair< T1, T2 > &`value`, ::std::ostream *os)
- template<typename T>
 void `UniversalPrintArray` (const T *begin, size_t len, ::std::ostream *os)
- `GTEST_API_ void UniversalPrintArray` (const char *begin, size_t len, ::std::ostream *os)
- `GTEST_API_ void UniversalPrintArray` (const wchar_t *begin, size_t len, ::std::ostream *os)
- template<typename T>
 void `UniversalTersePrint` (const T &`value`, ::std::ostream *os)
- template<typename T1, typename T2>
 AssertionResult `CmpHelperEQFailure` (const char *lhs_expression, const char *rhs_expression, const T1 &lhs, const T2 &rhs)

- template<typename T1, typename T2>
AssertionResult **CmpHelperEQ** (const char *lhs_expression, const char *rhs_expression, const T1 &lhs, const T2 &rhs)
- **GTEST_API_ AssertionResult CmpHelperEQ** (const char *lhs_expression, const char *rhs_expression, BiggestInt lhs, BiggestInt rhs)
- template<typename T1, typename T2>
AssertionResult **CmpHelperOpFailure** (const char *expr1, const char *expr2, const T1 &val1, const T2 &val2, const char *op)
- **GTEST_IMPL_CMP_HELPER_(NE, !=)**
- **GTEST_IMPL_CMP_HELPER_(LE, <=)**
- **GTEST_IMPL_CMP_HELPER_(LT, <)**
- **GTEST_IMPL_CMP_HELPER_(GE, >=)**
- **GTEST_IMPL_CMP_HELPER_(GT, >)**
- **GTEST_API_ AssertionResult CmpHelperSTREQ** (const char *s1_expression, const char *s2_expression, const char *s1, const char *s2)
- **GTEST_API_ AssertionResult CmpHelperSTRCASEEQ** (const char *s1_expression, const char *s2_left_expression, const char *s1, const char *s2)
- **GTEST_API_ AssertionResult CmpHelperSTRNE** (const char *s1_expression, const char *s2_expression, const char *s1, const char *s2)
- **GTEST_API_ AssertionResult CmpHelperSTRCASENE** (const char *s1_expression, const char *s2_left_expression, const char *s1, const char *s2)
- **GTEST_API_ AssertionResult CmpHelperSTREQ** (const char *s1_expression, const char *s2_expression, const wchar_t *s1, const wchar_t *s2)
- **GTEST_API_ AssertionResult CmpHelperSTRNE** (const char *s1_expression, const char *s2_expression, const wchar_t *s1, const wchar_t *s2)
- template<typename RawType>
AssertionResult **CmpHelperFloatingPointEQ** (const char *lhs_expression, const char *rhs_expression, RawType lhs_value, RawType rhs_value)
- **GTEST_API_ AssertionResult DoubleNearPredFormat** (const char *expr1, const char *expr2, const char *abs_error_expr, double val1, double val2, double abs_error)
- **GTEST_DECLARE_string_(internal_run_death_test)**
- char **IsNullLiteralHelper** (Secret *p)
- char(& IsNullLiteralHelper (...))[2]
- **GTEST_API_ std::string AppendUserMessage** (const std::string >est_msg, const Message &user_msg)
- **GTEST_API_ std::string DiffStrings** (const std::string &left, const std::string &right, size_t *total_line_count)
- **GTEST_API_ AssertionResult EqFailure** (const char *expected_expression, const char *actual_expression, const std::string &expected_value, const std::string &actual_value, bool ignoring_case)
- **GTEST_API_ std::string GetBoolAssertionFailureMessage** (const AssertionResult &assertion_result, const char *expression_text, const char *actual_predicate_value, const char *expected_predicate_value)
- template<typename T>
TypId GetTypId()
- **GTEST_API_ TypId GetTestTypId()**
- **GTEST_API_ TestInfo * MakeAndRegisterTestInfo** (const char *test_case_name, const char *name, const char *type_param, const char *value_param, CodeLocation code_location, TypId fixture_class_id, SetUpTestCaseFunc set_up_tc, TearDownTestCaseFunc tear_down_tc, TestFactoryBase *factory)
- **GTEST_API_ bool SkipPrefix** (const char *prefix, const char **pstr)
- **GTEST_API_ std::string GetCurrentOsStackTraceExceptTop** (UnitTest *unit_test, int skip_count)
- **GTEST_API_ bool AlwaysTrue()**
- bool **AlwaysFalse()**
- template<class C>
IsContainer IsContainerTest (int, typename C::iterator *=NULL, typename C::const_iterator *=NULL)
- template<class C>
 IsNotContainer IsContainerTest (long)
- template<typename T, typename U>
bool **ArrayEq** (const T *lhs, size_t size, const U *rhs)

- template<typename T, typename U>
bool **ArrayEq** (const T &lhs, const U &rhs)
- template<typename T, typename U, size_t N>
bool **ArrayEq** (const T(&lhs)[N], const U(&rhs)[N])
- template<typename Iter, typename Element>
Iter **ArrayAwareFind** (Iter begin, Iter end, const Element &elem)
- template<typename T, typename U>
void **CopyArray** (const T *from, size_t size, U *to)
- template<typename T, typename U>
void **CopyArray** (const T &from, U *to)
- template<typename T, typename U, size_t N>
void **CopyArray** (const T(&from)[N], U(*to)[N])
- **GTEST_API_GTEST_DECLARE_STATIC_MUTEX_** (g_linked_ptr_mutex)
- template<typename T>
bool **operator==** (T *ptr, const linked_ptr< T > &x)
- template<typename T>
bool **operator!=** (T *ptr, const linked_ptr< T > &x)
- template<typename T>
linked_ptr< T > **make_linked_ptr** (T *ptr)
- **GTEST_API_void ReportInvalidTestCaseType** (const char *test_case_name, CodeLocation code_location)
- template<class ParamType>
std::string **DefaultParamName** (const TestParamInfo< ParamType > &info)
- template<class ParamType, class ParamNameGenFunctor>
ParamNameGenFunctor **GetParamNameGen** (ParamNameGenFunctor func)
- template<class ParamType>
ParamNameGenFunc< ParamType >::Type * **GetParamNameGen** ()
- **GTEST_API_bool IsTrue** (bool condition)
- **GTEST_API::std::string FormatFileLocation** (const char *file, int line)
- **GTEST_API::std::string FormatCompilerIndependentFileLocation** (const char *file, int line)
- void **LogToStderr** ()
- void **FlushInfoLog** ()
- template<typename T>
const T & **move** (const T &t)
- template<typename T>
GTEST_ADD_REFERENCE_ (T) forward(GTEST_ADD_REFERENCE_(T) t)
- template<typename To>
To **ImplicitCast_** (To x)
- template<typename To, typename From>
To **DownCast_** (From *f)
- template<class Derived, class Base>
Derived * **CheckedDowncastToActualType** (Base *base)
- **GTEST_API_void CaptureStdout** ()
- **GTEST_API_std::string GetCapturedStdout** ()
- **GTEST_API_void CaptureStderr** ()
- **GTEST_API_std::string GetCapturedStderr** ()
- **GTEST_API_size_t GetFileSize** (FILE *file)
- **GTEST_API_std::string ReadEntireFile** (FILE *file)
- **GTEST_API_std::vector< std::string > GetArgs** ()
- **GTEST_API_size_t GetThreadCount** ()
- bool **IsAlpha** (char ch)
- bool **IsAINum** (char ch)
- bool **IsDigit** (char ch)
- bool **IsLower** (char ch)
- bool **IsSpace** (char ch)
- bool **IsUpper** (char ch)
- bool **IsXDigit** (char ch)

- bool `IsXDigit` (wchar_t ch)
- char `ToLower` (char ch)
- char `ToUpper` (char ch)
- std::string `StripTrailingSpaces` (std::string str)
- bool `ParseInt32` (const `Message` &src_text, const char *str, Int32 *value)
- bool `BoolFromGTestEnv` (const char *flag, bool default_val)
- `GTEST_API_ Int32 Int32FromGTestEnv` (const char *flag, Int32 default_val)
- std::string `OutputFlagAlsoCheckEnvVar` ()
- const char * `StringFromGTestEnv` (const char *flag, const char *default_val)
- `GTEST_API_ std::string StringStreamToString` (::std::stringstream *stream)
- std::string `CanonicalizeForStdLibVersioning` (std::string s)
- template<typename T>
std::string `GetTypeName` ()

Zmienne

- const char `kDeathTestStyleFlag` [] = "death_test_style"
- const char `kDeathTestUseFork` [] = "death_test_use_fork"
- const char `kInternalRunDeathTestFlag` [] = "internal_run_death_test"
- `GTEST_API_ const char kStackTraceMarker` []
- template<typename T>
bool `TypeIdHelper< T >::dummy` = false
- template<typename From, typename To>
const bool `ImplicitlyConvertible< From, To >::value`
- template<typename T>
const bool `IsHashTable< T >::value`
- template<bool bool_value>
const bool `bool_constant< bool_value >::value`
- const `BiggestInt kMaxBiggestInt`

7.6.1 Dokumentacja definicji typów

7.6.1.1 BiggestInt

```
typedef long long testing::internal::BiggestInt
```

7.6.1.2 Double

```
typedef FloatingPoint<double> testing::internal::Double
```

7.6.1.3 false_type

```
typedef bool_constant<false> testing::internal::false_type
```

7.6.1.4 Float

```
typedef FloatingPoint<float> testing::internal::Float
```

7.6.1.5 **Int32**

```
typedef TypeWithSize<4>::Int testing::internal::Int32
```

7.6.1.6 **Int64**

```
typedef TypeWithSize<8>::Int testing::internal::Int64
```

7.6.1.7 **IsContainer**

```
typedef int testing::internal::IsContainer
```

7.6.1.8 **IsNotContainer**

```
typedef char testing::internal::IsNotContainer
```

7.6.1.9 **MutexLock**

```
typedef GTestMutexLock testing::internal::MutexLock
```

7.6.1.10 **SetUpTestCaseFunc**

```
typedef void(* testing::internal::SetUpTestCaseFunc) ()
```

7.6.1.11 **string**

```
typedef ::std::string testing::internal::string
```

7.6.1.12 **Strings**

```
typedef ::std::vector< ::std::string> testing::internal::Strings
```

7.6.1.13 **TearDownTestCaseFunc**

```
typedef void(* testing::internal::TearDownTestCaseFunc) ()
```

7.6.1.14 **TimeInMillis**

```
typedef TypeWithSize<8>::Int testing::internal::TimeInMillis
```

7.6.1.15 true_type

```
typedef bool_constant<true> testing::internal::true_type
```

7.6.1.16 TypeId

```
typedef const void* testing::internal::TypeId
```

7.6.1.17 UInt32

```
typedef TypeWithSize<4>::UInt testing::internal::UInt32
```

7.6.1.18 UInt64

```
typedef TypeWithSize<8>::UInt testing::internal::UInt64
```

7.6.1.19 wstring

```
typedef ::std::wstring testing::internal::wstring
```

7.6.2 Dokumentacja typów wyliczanych

7.6.2.1 DefaultPrinterType

```
enum testing::internal::DefaultPrinterType
```

Wartości wyliczeń

kPrintContainer	
kPrintPointer	
kPrintFunctionPointer	
kPrintOther	

7.6.2.2 GTestLogSeverity

```
enum testing::internal::GTestLogSeverity
```

Wartości wyliczeń

GTEST_INFO	
GTEST_WARNING	

GTEST_ERROR	
GTEST_FATAL	

7.6.3 Dokumentacja funkcji

7.6.3.1 AlwaysFalse()

```
bool testing::internal::AlwaysFalse () [inline]
```

7.6.3.2 AlwaysTrue()

```
GTEST_API_ bool testing::internal::AlwaysTrue ()
```

7.6.3.3 AppendUserMessage()

```
GTEST_API_ std::string testing::internal::AppendUserMessage (
    const std::string & gtest_msg,
    const Message & user_msg)
```

7.6.3.4 ArrayAwareFind()

```
template<typename Iter, typename Element>
Iter testing::internal::ArrayAwareFind (
    Iter begin,
    Iter end,
    const Element & elem)
```

7.6.3.5 ArrayEq() [1/3]

```
template<typename T, typename U>
bool testing::internal::ArrayEq (
    const T & lhs,
    const U & rhs) [inline]
```

7.6.3.6 ArrayEq() [2/3]

```
template<typename T, typename U>
bool testing::internal::ArrayEq (
    const T * lhs,
    size_t size,
    const U * rhs)
```

7.6.3.7 **ArrayEq()** [3/3]

```
template<typename T, typename U, size_t N>
bool testing::internal::ArrayEq (
    const T(&) lhs[N],
    const U(&) rhs[N]) [inline]
```

7.6.3.8 **BoolFromGTestEnv()**

```
bool testing::internal::BoolFromGTestEnv (
    const char * flag,
    bool default_val)
```

7.6.3.9 **CanonicalizeForStdLibVersioning()**

```
std::string testing::internal::CanonicalizeForStdLibVersioning (
    std::string s) [inline]
```

7.6.3.10 **CaptureStderr()**

```
GTEST_API_ void testing::internal::CaptureStderr ()
```

7.6.3.11 **CaptureStdout()**

```
GTEST_API_ void testing::internal::CaptureStdout ()
```

7.6.3.12 **CheckedDowncastToActualType()**

```
template<class Derived, class Base>
Derived * testing::internal::CheckedDowncastToActualType (
    Base * base)
```

7.6.3.13 **CmpHelperEQ()** [1/2]

```
GTEST_API_ AssertionResult testing::internal::CmpHelperEQ (
    const char * lhs_expression,
    const char * rhs_expression,
    BiggestInt lhs,
    BiggestInt rhs)
```

7.6.3.14 **CmpHelperEQ()** [2/2]

```
template<typename T1, typename T2>
AssertionResult testing::internal::CmpHelperEQ (
    const char * lhs_expression,
    const char * rhs_expression,
    const T1 & lhs,
    const T2 & rhs)
```

7.6.3.15 CmpHelperEQFailure()

```
template<typename T1, typename T2>
AssertionResult testing::internal::CmpHelperEQFailure (
    const char * lhs_expression,
    const char * rhs_expression,
    const T1 & lhs,
    const T2 & rhs)
```

7.6.3.16 CmpHelperFloatingPointEQ()

```
template<typename RawType>
AssertionResult testing::internal::CmpHelperFloatingPointEQ (
    const char * lhs_expression,
    const char * rhs_expression,
    RawType lhs_value,
    RawType rhs_value)
```

7.6.3.17 CmpHelperOpFailure()

```
template<typename T1, typename T2>
AssertionResult testing::internal::CmpHelperOpFailure (
    const char * expr1,
    const char * expr2,
    const T1 & val1,
    const T2 & val2,
    const char * op)
```

7.6.3.18 CmpHelperSTRCASEEQ()

```
GTEST_API_ AssertionResult testing::internal::CmpHelperSTRCASEEQ (
    const char * s1_expression,
    const char * s2_expression,
    const char * s1,
    const char * s2)
```

7.6.3.19 CmpHelperSTRCASENE()

```
GTEST_API_ AssertionResult testing::internal::CmpHelperSTRCASENE (
    const char * s1_expression,
    const char * s2_expression,
    const char * s1,
    const char * s2)
```

7.6.3.20 CmpHelperSTREQ() [1/2]

```
GTEST_API_ AssertionResult testing::internal::CmpHelperSTREQ (
    const char * s1_expression,
    const char * s2_expression,
    const char * s1,
    const char * s2)
```

7.6.3.21 CmpHelperSTREQ() [2/2]

```
GTEST_API_ AssertionResult testing::internal::CmpHelperSTREQ (
    const char * s1_expression,
    const char * s2_expression,
    const wchar_t * s1,
    const wchar_t * s2)
```

7.6.3.22 CmpHelperSTRNE() [1/2]

```
GTEST_API_ AssertionResult testing::internal::CmpHelperSTRNE (
    const char * s1_expression,
    const char * s2_expression,
    const char * s1,
    const char * s2)
```

7.6.3.23 CmpHelperSTRNE() [2/2]

```
GTEST_API_ AssertionResult testing::internal::CmpHelperSTRNE (
    const char * s1_expression,
    const char * s2_expression,
    const wchar_t * s1,
    const wchar_t * s2)
```

7.6.3.24 CopyArray() [1/3]

```
template<typename T, typename U>
void testing::internal::CopyArray (
    const T & from,
    U * to) [inline]
```

7.6.3.25 CopyArray() [2/3]

```
template<typename T, typename U>
void testing::internal::CopyArray (
    const T * from,
    size_t size,
    U * to)
```

7.6.3.26 CopyArray() [3/3]

```
template<typename T, typename U, size_t N>
void testing::internal::CopyArray (
    const T(&) from[N],
    U(*) to[N]) [inline]
```

7.6.3.27 DefaultParamName()

```
template<class ParamType>
std::string testing::internal::DefaultParamName (
    const TestParamInfo< ParamType > & info)
```

7.6.3.28 DefaultPrintTo() [1/4]

```
template<typename C>
void testing::internal::DefaultPrintTo (
    WrapPrinterType< kPrintContainer > ,
    const C & container,
    ::std::ostream * os)
```

7.6.3.29 DefaultPrintTo() [2/4]

```
template<typename T>
void testing::internal::DefaultPrintTo (
    WrapPrinterType< kPrintFunctionPointer > ,
    T * p,
    ::std::ostream * os)
```

7.6.3.30 DefaultPrintTo() [3/4]

```
template<typename T>
void testing::internal::DefaultPrintTo (
    WrapPrinterType< kPrintOther > ,
    const T & value,
    ::std::ostream * os)
```

7.6.3.31 DefaultPrintTo() [4/4]

```
template<typename T>
void testing::internal::DefaultPrintTo (
    WrapPrinterType< kPrintPointer > ,
    T * p,
    ::std::ostream * os)
```

7.6.3.32 DiffStrings()

```
GTEST_API_ std::string testing::internal::DiffStrings (
    const std::string & left,
    const std::string & right,
    size_t * total_line_count)
```

7.6.3.33 DoubleNearPredFormat()

```
GTEST_API_ AssertionResult testing::internal::DoubleNearPredFormat (
    const char * expr1,
    const char * expr2,
    const char * abs_error_expr,
    double val1,
    double val2,
    double abs_error)
```

7.6.3.34 DownCast_()

```
template<typename To, typename From>
To testing::internal::DownCast_ (
    From * f) [inline]
```

7.6.3.35 EqFailure()

```
GTEST_API_ AssertionResult testing::internal::EqFailure (
    const char * expected_expression,
    const char * actual_expression,
    const std::string & expected_value,
    const std::string & actual_value,
    bool ignoring_case)
```

7.6.3.36 FlushInfoLog()

```
void testing::internal::FlushInfoLog () [inline]
```

7.6.3.37 FormatCompilerIndependentFileLocation()

```
GTEST_API_::std::string testing::internal::FormatCompilerIndependentFileLocation (
    const char * file,
    int line)
```

7.6.3.38 FormatFileLocation()

```
GTEST_API_::std::string testing::internal::FormatFileLocation (
    const char * file,
    int line)
```

7.6.3.39 FormatForComparisonFailureMessage()

```
template<typename T1, typename T2>
std::string testing::internal::FormatForComparisonFailureMessage (
    const T1 & value,
    const T2 & )
```

7.6.3.40 GetArgs()

```
GTEST_API_ std::vector< std::string > testing::internal::GetArgs ()
```

7.6.3.41 GetBoolAssertionFailureMessage()

```
GTEST_API_ std::string testing::internal::GetBoolAssertionFailureMessage ( const AssertionResult & assertion_result, const char * expression_text, const char * actual_predicate_value, const char * expected_predicate_value)
```

7.6.3.42 GetCapturedStderr()

```
GTEST_API_ std::string testing::internal::GetCapturedStderr ()
```

7.6.3.43 GetCapturedStdout()

```
GTEST_API_ std::string testing::internal::GetCapturedStdout ()
```

7.6.3.44 GetCurrentOsStackTraceExceptTop()

```
GTEST_API_ std::string testing::internal::GetCurrentOsStackTraceExceptTop ( UnitTest * unit_test, int skip_count)
```

7.6.3.45 GetFileSize()

```
GTEST_API_ size_t testing::internal::GetFileSize ( FILE * file)
```

7.6.3.46 GetParamNameGen() [1/2]

```
template<class ParamType>
ParamNameGenFunc< ParamType >::Type * testing::internal::GetParamNameGen ()
```

7.6.3.47 GetParamNameGen() [2/2]

```
template<class ParamType, class ParamNameGenFunctor>
ParamNameGenFunctor testing::internal::GetParamNameGen ( ParamNameGenFunctor func)
```

7.6.3.48 GetTestId()

```
GTEST_API_ TypeId testing::internal::GetTestId ()
```

7.6.3.49 GetThreadCount()

```
GTEST_API_ size_t testing::internal::GetThreadCount ()
```

7.6.3.50 GetTypeId()

```
template<typename T>
TypeId testing::internal::GetTypeId ()
```

7.6.3.51 GetTypeName()

```
template<typename T>
std::string testing::internal::GetTypeName ()
```

7.6.3.52 GTEST_ADD_REFERENCE_()

```
template<typename T>
testing::internal::GTEST_ADD_REFERENCE_ (
    T )
```

7.6.3.53 GTEST_DECLARE_STATIC_MUTEX_()

```
GTEST_API_ testing::internal::GTEST_DECLARE_STATIC_MUTEX_ (
    g_linked_ptr_mutex )
```

7.6.3.54 GTEST_DECLARE_string_()

```
testing::internal::GTEST_DECLARE_string_ (
    internal_run_death_test )
```

7.6.3.55 GTEST_IMPL_CMP_HELPER_() [1/5]

```
testing::internal::GTEST_IMPL_CMP_HELPER_ (
    GE ,
    >= )
```

7.6.3.56 GTEST_IMPL_CMP_HELPER_() [2/5]

```
testing::internal::GTEST_IMPL_CMP_HELPER_ (
    GT )
```

7.6.3.57 GTEST_IMPL_CMP_HELPER_() [3/5]

```
testing::internal::GTEST_IMPL_CMP_HELPER_ (
    LE ,
    <= )
```

7.6.3.58 GTEST_IMPL_CMP_HELPER_() [4/5]

```
testing::internal::GTEST_IMPL_CMP_HELPER_ (
    LT )
```

7.6.3.59 GTEST_IMPL_CMP_HELPER_() [5/5]

```
testing::internal::GTEST_IMPL_CMP_HELPER_ (
    NE ,
    ! )
```

7.6.3.60 GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_() [1/2]

```
testing::internal::GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_ (
    char )
```

7.6.3.61 GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_() [2/2]

```
testing::internal::GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_ (
    wchar_t )
```

7.6.3.62 GTEST_IMPL_FORMAT_C_STRING_AS_STRING_()

```
testing::internal::GTEST_IMPL_FORMAT_C_STRING_AS_STRING_ (
    char ,
    ::std::string )
```

7.6.3.63 ImplicitCast_()

```
template<typename To>
To testing::internal::ImplicitCast_ (
    To x) [inline]
```

7.6.3.64 Int32FromGTestEnv()

```
GTEST_API_ Int32 testing::internal::Int32FromGTestEnv (
    const char * flag,
    Int32 default_val)
```

7.6.3.65 IsAlNum()

```
bool testing::internal::IsAlNum (
    char ch) [inline]
```

7.6.3.66 IsAlpha()

```
bool testing::internal::IsAlpha (
    char ch) [inline]
```

7.6.3.67 IsContainerTest() [1/2]

```
template<class C>
IsContainer testing::internal::IsContainerTest (
    int ,
    typename C::iterator * = NULL,
    typename C::const_iterator * = NULL)
```

7.6.3.68 IsContainerTest() [2/2]

```
template<class C>
IsNotContainer testing::internal::IsContainerTest (
    long )
```

7.6.3.69 IsDigit()

```
bool testing::internal::IsDigit (
    char ch) [inline]
```

7.6.3.70 IsLower()

```
bool testing::internal::IsLower (
    char ch) [inline]
```

7.6.3.71 IsNullLiteralHelper() [1/2]

```
char(& testing::internal::IsNullLiteralHelper (
    ...)) [2]
```

7.6.3.72 IsNullLiteralHelper() [2/2]

```
char testing::internal::IsNullLiteralHelper (
    Secret * p)
```

7.6.3.73 IsSpace()

```
bool testing::internal::IsSpace (
    char ch) [inline]
```

7.6.3.74 IsTrue()

```
GTEST_API_ bool testing::internal::IsTrue (
    bool condition)
```

7.6.3.75 IsUpper()

```
bool testing::internal::IsUpper (
    char ch) [inline]
```

7.6.3.76 IsXDigit() [1/2]

```
bool testing::internal::IsXDigit (
    char ch) [inline]
```

7.6.3.77 IsXDigit() [2/2]

```
bool testing::internal::IsXDigit (
    wchar_t ch) [inline]
```

7.6.3.78 LogToStderr()

```
void testing::internal::LogToStderr () [inline]
```

7.6.3.79 make_linked_ptr()

```
template<typename T>
linked_ptr<T> testing::internal::make_linked_ptr (
    T * ptr)
```

7.6.3.80 MakeAndRegisterTestInfo()

```
GTEST_API_ TestInfo * testing::internal::MakeAndRegisterTestInfo (
    const char * test_case_name,
    const char * name,
    const char * type_param,
    const char * value_param,
    CodeLocation code_location,
    TypeId fixture_class_id,
    SetUpTestCaseFunc set_up_tc,
    TearDownTestCaseFunc tear_down_tc,
    TestFactoryBase * factory)
```

7.6.3.81 move()

```
template<typename T>
const T & testing::internal::move (
    const T & t)
```

7.6.3.82 operator"!=()

```
template<typename T>
bool testing::internal::operator!= (
    T * ptr,
    const linked_ptr< T > & x) [inline]
```

7.6.3.83 operator==()

```
template<typename T>
bool testing::internal::operator== (
    T * ptr,
    const linked_ptr< T > & x) [inline]
```

7.6.3.84 OutputFlagAlsoCheckEnvVar()

```
std::string testing::internal::OutputFlagAlsoCheckEnvVar ()
```

7.6.3.85 ParseInt32()

```
bool testing::internal::ParseInt32 (
    const Message & src_text,
    const char * str,
    Int32 * value)
```

7.6.3.86 PrintRawArrayTo()

```
template<typename T>
void testing::internal::PrintRawArrayTo (
    const T a[],
    size_t count,
    ::std::ostream * os)
```

7.6.3.87 PrintStringTo()

```
GTEST_API_ void testing::internal::PrintStringTo (
    const ::std::string & s,
    ::std::ostream * os)
```

7.6.3.88 PrintTo() [1/16]

```
void testing::internal::PrintTo (
    bool x,
    ::std::ostream * os)  [inline]
```

7.6.3.89 PrintTo() [2/16]

```
void testing::internal::PrintTo (
    char * s,
    ::std::ostream * os)  [inline]
```

7.6.3.90 PrintTo() [3/16]

```
void testing::internal::PrintTo (
    char c,
    ::std::ostream * os)  [inline]
```

7.6.3.91 PrintTo() [4/16]

```
template<typename T1, typename T2>
void testing::internal::PrintTo (
    const ::std::pair< T1, T2 > & value,
    ::std::ostream * os)
```

7.6.3.92 PrintTo() [5/16]

```
void testing::internal::PrintTo (
    const ::std::string & s,
    ::std::ostream * os)  [inline]
```

7.6.3.93 PrintTo() [6/16]

```
GTEST_API_ void testing::internal::PrintTo (
    const char * s,
    ::std::ostream * os)
```

7.6.3.94 PrintTo() [7/16]

```
void testing::internal::PrintTo (
    const signed char * s,
    ::std::ostream * os)  [inline]
```

7.6.3.95 PrintTo() [8/16]

```
template<typename T>
void testing::internal::PrintTo (
    const T & value,
    ::std::ostream * os)
```

7.6.3.96 PrintTo() [9/16]

```
void testing::internal::PrintTo (
    const unsigned char * s,
    ::std::ostream * os)  [inline]
```

7.6.3.97 PrintTo() [10/16]

```
GTEST_API_ void testing::internal::PrintTo (
    const wchar_t * s,
    ::std::ostream * os)
```

7.6.3.98 PrintTo() [11/16]

```
void testing::internal::PrintTo (
    signed char * s,
    ::std::ostream * os)  [inline]
```

7.6.3.99 PrintTo() [12/16]

```
GTEST_API_ void testing::internal::PrintTo (
    signed char c,
    ::std::ostream * os)
```

7.6.3.100 PrintTo() [13/16]

```
void testing::internal::PrintTo (
    unsigned char * s,
    ::std::ostream * os)  [inline]
```

7.6.3.101 PrintTo() [14/16]

```
GTEST_API_ void testing::internal::PrintTo (
    unsigned char c,
    ::std::ostream * os)
```

7.6.3.102 PrintTo() [15/16]

```
void testing::internal::PrintTo (
    wchar_t * s,
    ::std::ostream * os)  [inline]
```

7.6.3.103 PrintTo() [16/16]

```
GTEST_API_ void testing::internal::PrintTo (
    wchar_t wc,
    ::std::ostream * os)
```

7.6.3.104 ReadEntireFile()

```
GTEST_API_ std::string testing::internal::ReadEntireFile (
    FILE * file)
```

7.6.3.105 ReportInvalidTestCaseType()

```
GTEST_API_ void testing::internal::ReportInvalidTestCaseType (
    const char * test_case_name,
    CodeLocation code_location)
```

7.6.3.106 SkipPrefix()

```
GTEST_API_ bool testing::internal::SkipPrefix (
    const char * prefix,
    const char ** pstr)
```

7.6.3.107 StreamableToString()

```
template<typename T>
std::string testing::internal::StreamableToString (
    const T & streamable)
```

7.6.3.108 StringFromGTestEnv()

```
const char * testing::internal::StringFromGTestEnv (
    const char * flag,
    const char * default_val)
```

7.6.3.109 StringStreamToString()

```
GTEST_API_ std::string testing::internal::StringStreamToString (
    ::std::stringstream * stream)
```

7.6.3.110 StripTrailingSpaces()

```
std::string testing::internal::StripTrailingSpaces (
    std::string str) [inline]
```

7.6.3.111 ToLower()

```
char testing::internal::ToLower (
    char ch) [inline]
```

7.6.3.112 ToUpper()

```
char testing::internal::ToUpper (
    char ch) [inline]
```

7.6.3.113 UniversalPrint()

```
template<typename T>
void testing::internal::UniversalPrint (
    const T & value,
    ::std::ostream * os)
```

7.6.3.114 UniversalPrintArray() [1/3]

```
GTEST_API_ void testing::internal::UniversalPrintArray (
    const char * begin,
    size_t len,
    ::std::ostream * os)
```

7.6.3.115 UniversalPrintArray() [2/3]

```
template<typename T>
void testing::internal::UniversalPrintArray (
    const T * begin,
    size_t len,
    ::std::ostream * os)
```

7.6.3.116 UniversalPrintArray() [3/3]

```
GTEST_API_ void testing::internal::UniversalPrintArray (
    const wchar_t * begin,
    size_t len,
    ::std::ostream * os)
```

7.6.3.117 UniversalTersePrint()

```
template<typename T>
void testing::internal::UniversalTersePrint (
    const T & value,
    ::std::ostream * os)
```

7.6.4 Dokumentacja zmiennych

7.6.4.1 bool_constant< bool_value >::value

```
template<bool bool_value>
const bool testing::internal::bool_constant< bool_value >::value
```

7.6.4.2 ImplicitlyConvertible< From, To >::value

```
template<typename From, typename To>
const bool testing::internal::ImplicitlyConvertible< From, To >::value
```

7.6.4.3 IsHashTable< T >::value

```
template<typename T>
const bool testing::internal::IsHashTable< T >::value
```

7.6.4.4 kDeathTestStyleFlag

```
const char testing::internal::kDeathTestStyleFlag[] = "death_test_style"
```

7.6.4.5 kDeathTestUseFork

```
const char testing::internal::kDeathTestUseFork[] = "death_test_use_fork"
```

7.6.4.6 kInternalRunDeathTestFlag

```
const char testing::internal::kInternalRunDeathTestFlag[] = "internal_run_death_test"
```

7.6.4.7 kMax BiggestInt

```
const BiggestInt testing::internal::kMax BiggestInt
```

Wartość początkowa:

```
=
~(static_cast<BiggestInt>(1) << (8*sizeof(BiggestInt) - 1))
```

7.6.4.8 kStackTraceMarker

```
GTEST_API_ const char testing::internal::kStackTraceMarker[ ] [extern]
```

7.6.4.9 TypeIdHelper< T >::dummy_

```
template<typename T>
bool testing::internal::TypeIdHelper< T >::dummy_ = false
```

7.7 Dokumentacja przestrzeni nazw testing::internal2

Komponenty

- class [TypeWithoutFormatter](#)
- class [TypeWithoutFormatter< T, kProtobuf >](#)
- class [TypeWithoutFormatter< T, kConvertibleToInteger >](#)

Wyliczenia

- enum [TypeKind { kProtobuf, kConvertibleToInteger, kOtherType }](#)

Funkcje

- [GTEST_API_ void PrintBytesInObjectTo \(const unsigned char *obj_bytes, size_t count, ::std::ostream *os\)](#)
- template<typename Char, typename CharTraits, typename T>
`::std::basic_ostream< Char, CharTraits > & operator<< (::std::basic_ostream< Char, CharTraits > &os, const T &x)`

Zmienne

- const size_t [kProtobufOneLinerMaxLength](#) = 50

7.7.1 Dokumentacja typów wyliczanych

7.7.1.1 TypeKind

```
enum testing::internal2::TypeKind
```

Wartości wyliczeń

kProtobuf	
kConvertibleToInteger	
kOtherType	

7.7.2 Dokumentacja funkcji

7.7.2.1 operator<<()

```
template<typename Char, typename CharTraits, typename T>
::std::basic_ostream< Char, CharTraits > & testing::internal2::operator<< (
    ::std::basic_ostream< Char, CharTraits > & os,
    const T & x)
```

7.7.2.2 PrintBytesInObjectTo()

```
GTEST_API_ void testing::internal2::PrintBytesInObjectTo (
    const unsigned char * obj_bytes,
    size_t count,
    ::std::ostream * os)
```

7.7.3 Dokumentacja zmiennych

7.7.3.1 kProtobufOneLinerMaxLength

```
const size_t testing::internal2::kProtobufOneLinerMaxLength = 50
```

7.8 Dokumentacja przestrzeni nazw testing::internal::edit_distance

Wyliczenia

- enum `EditType` { `kMatch` , `kAdd` , `kRemove` , `kReplace` }

Funkcje

- `GTEST_API_ std::vector< EditType > CalculateOptimalEdits (const std::vector< size_t > &left, const std::vector< size_t > &right)`
- `GTEST_API_ std::vector< EditType > CalculateOptimalEdits (const std::vector< std::string > &left, const std::vector< std::string > &right)`
- `GTEST_API_ std::string CreateUnifiedDiff (const std::vector< std::string > &left, const std::vector< std::string > &right, size_t context=2)`

7.8.1 Dokumentacja typów wyliczanych

7.8.1.1 EditType

```
enum testing::internal::edit_distance::EditType
```

Wartości wyliczeń

kMatch	
kAdd	
kRemove	
kReplace	

7.8.2 Dokumentacja funkcji

7.8.2.1 CalculateOptimalEdits() [1/2]

```
GTEST_API_ std::vector< EditType > testing::internal::edit_distance::CalculateOptimalEdits (
    const std::vector< size_t > & left,
    const std::vector< size_t > & right)
```

7.8.2.2 CalculateOptimalEdits() [2/2]

```
GTEST_API_ std::vector< EditType > testing::internal::edit_distance::CalculateOptimalEdits (
    const std::vector< std::string > & left,
    const std::vector< std::string > & right)
```

7.8.2.3 CreateUnifiedDiff()

```
GTEST_API_ std::string testing::internal::edit_distance::CreateUnifiedDiff (
    const std::vector< std::string > & left,
    const std::vector< std::string > & right,
    size_t context = 2)
```

7.9 Dokumentacja przestrzeni nazw testing::internal::posix

Definicje typów

- `typedef struct stat StatStruct`

Funkcje

- `int FileNo (FILE *file)`
- `int IsATTY (int fd)`
- `int Stat (const char *path, StatStruct *buf)`
- `int StrCaseCmp (const char *s1, const char *s2)`
- `char * StrDup (const char *src)`
- `int RmDir (const char *dir)`
- `bool IsDir (const StatStruct &st)`
- `const char * StrNCpy (char *dest, const char *src, size_t n)`
- `int ChDir (const char *dir)`
- `FILE * FOpen (const char *path, const char *mode)`
- `FILE * FReopen (const char *path, const char *mode, FILE *stream)`
- `FILE * FDOpen (int fd, const char *mode)`
- `int FClose (FILE *fp)`
- `int Read (int fd, void *buf, unsigned int count)`
- `int Write (int fd, const void *buf, unsigned int count)`
- `int Close (int fd)`
- `const char * StrError (int errnum)`
- `const char * GetEnv (const char *name)`
- `void Abort ()`

7.9.1 Dokumentacja definicji typów

7.9.1.1 StatStruct

```
typedef struct stat testing::internal::posix::StatStruct
```

7.9.2 Dokumentacja funkcji

7.9.2.1 Abort()

```
void testing::internal::posix::Abort () [inline]
```

7.9.2.2 ChDir()

```
int testing::internal::posix::ChDir (
    const char * dir) [inline]
```

7.9.2.3 Close()

```
int testing::internal::posix::Close (
    int fd) [inline]
```

7.9.2.4 FClose()

```
int testing::internal::posix::FClose (
    FILE * fp) [inline]
```

7.9.2.5 FDOpen()

```
FILE * testing::internal::posix::FDOpen (
    int fd,
    const char * mode) [inline]
```

7.9.2.6 FileNo()

```
int testing::internal::posix::FileNo (
    FILE * file) [inline]
```

7.9.2.7 FOpen()

```
FILE * testing::internal::posix::FOpen (
    const char * path,
    const char * mode) [inline]
```

7.9.2.8 FReopen()

```
FILE * testing::internal::posix::FReopen (
    const char * path,
    const char * mode,
    FILE * stream) [inline]
```

7.9.2.9 GetEnv()

```
const char * testing::internal::posix::GetEnv (
    const char * name) [inline]
```

7.9.2.10 IsATTY()

```
int testing::internal::posix::IsATTY (
    int fd) [inline]
```

7.9.2.11 IsDir()

```
bool testing::internal::posix::IsDir (
    const StatStruct & st) [inline]
```

7.9.2.12 Read()

```
int testing::internal::posix::Read (
    int fd,
    void * buf,
    unsigned int count) [inline]
```

7.9.2.13 RmDir()

```
int testing::internal::posix::RmDir (
    const char * dir) [inline]
```

7.9.2.14 Stat()

```
int testing::internal::posix::Stat (
    const char * path,
    StatStruct * buf) [inline]
```

7.9.2.15 StrCaseCmp()

```
int testing::internal::posix::StrCaseCmp (
    const char * s1,
    const char * s2) [inline]
```

7.9.2.16 StrDup()

```
char * testing::internal::posix::StrDup (
    const char * src) [inline]
```

7.9.2.17 StrError()

```
const char * testing::internal::posix::StrError (
    int errnum) [inline]
```

7.9.2.18 StrNCpy()

```
const char * testing::internal::posix::StrNCpy (
    char * dest,
    const char * src,
    size_t n) [inline]
```

7.9.2.19 Write()

```
int testing::internal::posix::Write (
    int fd,
    const void * buf,
    unsigned int count) [inline]
```

7.10 Dokumentacja przestrzeni nazw testing_internal

Funkcje

- template<typename T>
void [DefaultPrintNonContainerTo](#) (const T &value, ::std::ostream *os)

7.10.1 Dokumentacja funkcji

7.10.1.1 DefaultPrintNonContainerTo()

```
template<typename T>
void testing_internal::DefaultPrintNonContainerTo (
    const T & value,
    ::std::ostream * os)
```

Rozdział 8

Dokumentacja klas

8.1 Dokumentacja szablonu struktury std::tr1::gtest_internal::AddRef< T >

```
#include <gtest-tuple.h>
```

Typy publiczne

- typedef T & [type](#)

8.1.1 Dokumentacja składowych definicji typu

8.1.1.1 type

```
template<typename T>
typedef T& std::tr1::gtest\_internal::AddRef< T >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-tuple.h](#)

8.2 Dokumentacja szablonu struktury std::tr1::gtest_internal::AddRef< T & >

```
#include <gtest-tuple.h>
```

Typy publiczne

- typedef T & [type](#)

8.2.1 Dokumentacja składowych definicji typu

8.2.1.1 type

```
template<typename T>
typedef T& std::tr1::gtest_internal::AddRef< T & >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.3 Dokumentacja szablonu struktury testing::internal::AddReference< T >

```
#include <gtest-port.h>
```

Typy publiczne

- typedef T & type

8.3.1 Dokumentacja składowych definicji typu

8.3.1.1 type

```
template<typename T>
typedef T& testing::internal::AddReference< T >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.4 Dokumentacja szablonu struktury testing::internal::AddReference< T & >

```
#include <gtest-port.h>
```

Typy publiczne

- typedef T & type

8.4.1 Dokumentacja składowych definicji typu

8.4.1.1 type

```
template<typename T>
typedef T& testing::internal::AddReference< T & >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.5 Dokumentacja klasy testing::internal::AssertHelper

```
#include <gtest.h>
```

Metody publiczne

- `AssertHelper` (`TestPartResult::Type type`, `const char *file`, `int line`, `const char *message`)
- `~AssertHelper` ()
- `void operator=` (`const Message &message`) const

8.5.1 Dokumentacja konstruktora i destruktora

8.5.1.1 AssertHelper()

```
testing::internal::AssertHelper::AssertHelper (
    TestPartResult::Type type,
    const char * file,
    int line,
    const char * message)
```

8.5.1.2 ~AssertHelper()

```
testing::internal::AssertHelper::~AssertHelper ()
```

8.5.2 Dokumentacja funkcji składowych

8.5.2.1 operator=()

```
void testing::internal::AssertHelper::operator= (
    const Message & message) const
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

8.6 Dokumentacja szablonu struktury testing::internal::bool_constant< bool_value >

```
#include <gtest-port.h>
```

Typy publiczne

- `typedef bool_constant< bool_value > type`

Statyczne atrybuty publiczne

- `static const bool value = bool_value`

8.6.1 Dokumentacja składowych definicji typu

8.6.1.1 type

```
template<bool bool_value>
typedef bool_constant<bool_value> testing::internal::bool_constant< bool_value >::type
```

8.6.2 Dokumentacja atrybutów składowych

8.6.2.1 value

```
template<bool bool_value>
const bool testing::internal::bool_constant< bool_value >::value = bool_value [static]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.7 Dokumentacja szablonu struktury std::tr1::gtest_internal::ByRef< T >

```
#include <gtest-tuple.h>
```

Typy publiczne

- `typedef const T & type`

8.7.1 Dokumentacja składowych definicji typu

8.7.1.1 type

```
template<typename T>
typedef const T& std::tr1::gtest_internal::ByRef< T >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.8 Dokumentacja szablonu struktury std::tr1::gtest_internal::ByRef< T & >

```
#include <gtest-tuple.h>
```

Typy publiczne

- typedef T & [type](#)

8.8.1 Dokumentacja składowych definicji typu

8.8.1.1 type

```
template<typename T>
typedef T& std::tr1::gtest_internal::ByRef< T & >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.9 Dokumentacja struktury testing::internal::CodeLocation

```
#include <gtest-internal.h>
```

Metody publiczne

- [CodeLocation](#) (const std::string &a_file, int a_line)

Atrybuty publiczne

- std::string [file](#)
- int [line](#)

8.9.1 Dokumentacja konstruktora i destruktora

8.9.1.1 **CodeLocation()**

```
testing::internal::CodeLocation::CodeLocation (
    const std::string & a_file,
    int a_line) [inline]
```

8.9.2 Dokumentacja atrybutów składowych

8.9.2.1 **file**

```
std::string testing::internal::CodeLocation::file
```

8.9.2.2 **line**

```
int testing::internal::CodeLocation::line
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.10 Dokumentacja szablonu struktury

testing::internal::CompileAssert< bool >

```
#include <gtest-port.h>
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.11 Dokumentacja szablonu struktury

testing::internal::CompileAssertTypesEqual< T1, T2 >

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.12 Dokumentacja szablonu struktury

testing::internal::CompileAssertTypesEqual< T, T >

```
#include <gtest-internal.h>
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.13 Dokumentacja struktury testing::internal::ConstCharPtr

```
#include <gtest-internal.h>
```

Metody publiczne

- [ConstCharPtr](#) (const char *str)
- [operator bool](#) () const

Atrybuty publiczne

- const char * [value](#)

8.13.1 Dokumentacja konstruktora i destruktora

8.13.1.1 ConstCharPtr()

```
testing::internal::ConstCharPtr::ConstCharPtr (
    const char * str) [inline]
```

8.13.2 Dokumentacja funkcji składowych

8.13.2.1 operator bool()

```
testing::internal::ConstCharPtr::operator bool () const [inline]
```

8.13.3 Dokumentacja atrybutów składowych

8.13.3.1 value

```
const char* testing::internal::ConstCharPtr::value
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-internal.h](#)

8.14 Dokumentacja szablonu struktury testing::internal::ConstRef< T >

```
#include <gtest-port.h>
```

Typy publiczne

- `typedef const T & type`

8.14.1 Dokumentacja składowych definicji typu

8.14.1.1 type

```
template<typename T>
typedef const T & testing::internal::ConstRef< T >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.15 Dokumentacja szablonu struktury testing::internal::ConstRef< T &

>

```
#include <gtest-port.h>
```

Typy publiczne

- `typedef T & type`

8.15.1 Dokumentacja składowych definicji typu

8.15.1.1 type

```
template<typename T>
typedef T & testing::internal::ConstRef< T & >::type
```

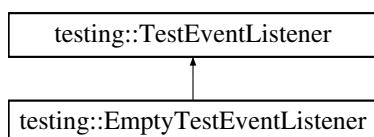
Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.16 Dokumentacja klasy testing::EmptyTestEventListener

```
#include <gtest.h>
```

Diagram dziedziczenia dla testing::EmptyTestEventListener



Metody publiczne

- virtual void `OnTestProgramStart` (const `UnitTest` &)
- virtual void `OnTestIterationStart` (const `UnitTest` &, int)
- virtual void `OnEnvironmentsSetUpStart` (const `UnitTest` &)
- virtual void `OnEnvironmentsSetUpEnd` (const `UnitTest` &)
- virtual void `OnTestCaseStart` (const `TestCase` &)
- virtual void `OnTestStart` (const `TestInfo` &)
- virtual void `OnTestPartResult` (const `TestPartResult` &)
- virtual void `OnTestEnd` (const `TestInfo` &)
- virtual void `OnTestCaseEnd` (const `TestCase` &)
- virtual void `OnEnvironmentsTearDownStart` (const `UnitTest` &)
- virtual void `OnEnvironmentsTearDownEnd` (const `UnitTest` &)
- virtual void `OnTestIterationEnd` (const `UnitTest` &, int)
- virtual void `OnTestProgramEnd` (const `UnitTest` &)

Metody publiczne dziedziczone z `testing::TestEventListener`

- virtual `~TestEventListener` ()

8.16.1 Dokumentacja funkcji składowych

8.16.1.1 `OnEnvironmentsSetUpEnd()`

```
virtual void testing::EmptyTestEventListener::OnEnvironmentsSetUpEnd (
    const UnitTest & ) [inline], [virtual]
```

Implementuje `testing::TestEventListener`.

8.16.1.2 `OnEnvironmentsSetUpStart()`

```
virtual void testing::EmptyTestEventListener::OnEnvironmentsSetUpStart (
    const UnitTest & ) [inline], [virtual]
```

Implementuje `testing::TestEventListener`.

8.16.1.3 `OnEnvironmentsTearDownEnd()`

```
virtual void testing::EmptyTestEventListener::OnEnvironmentsTearDownEnd (
    const UnitTest & ) [inline], [virtual]
```

Implementuje `testing::TestEventListener`.

8.16.1.4 `OnEnvironmentsTearDownStart()`

```
virtual void testing::EmptyTestEventListener::OnEnvironmentsTearDownStart (
    const UnitTest & ) [inline], [virtual]
```

Implementuje `testing::TestEventListener`.

8.16.1.5 OnTestCaseEnd()

```
virtual void testing::EmptyTestEventListener::OnTestCaseEnd (
    const TestCase & ) [inline], [virtual]
```

Implementuje [testing::TestEventListener](#).

8.16.1.6 OnTestCaseStart()

```
virtual void testing::EmptyTestEventListener::OnTestCaseStart (
    const TestCase & ) [inline], [virtual]
```

Implementuje [testing::TestEventListener](#).

8.16.1.7 OnTestEnd()

```
virtual void testing::EmptyTestEventListener::OnTestEnd (
    const TestInfo & ) [inline], [virtual]
```

Implementuje [testing::TestEventListener](#).

8.16.1.8 OnTestIterationEnd()

```
virtual void testing::EmptyTestEventListener::OnTestIterationEnd (
    const UnitTest & ,
    int ) [inline], [virtual]
```

Implementuje [testing::TestEventListener](#).

8.16.1.9 OnTestIterationStart()

```
virtual void testing::EmptyTestEventListener::OnTestIterationStart (
    const UnitTest & ,
    int ) [inline], [virtual]
```

Implementuje [testing::TestEventListener](#).

8.16.1.10 OnTestPartResult()

```
virtual void testing::EmptyTestEventListener::OnTestPartResult (
    const TestPartResult & ) [inline], [virtual]
```

Implementuje [testing::TestEventListener](#).

8.16.1.11 OnTestProgramEnd()

```
virtual void testing::EmptyTestEventListener::OnTestProgramEnd (
    const UnitTest & ) [inline], [virtual]
```

Implementuje [testing::TestEventListener](#).

8.16.1.12 OnTestProgramStart()

```
virtual void testing::EmptyTestEventListener::OnTestProgramStart (
    const UnitTest & ) [inline], [virtual]
```

Implementuje [testing::TestEventListener](#).

8.16.1.13 OnTestStart()

```
virtual void testing::EmptyTestEventListener::OnTestStart (
    const TestInfo & ) [inline], [virtual]
```

Implementuje [testing::TestEventListener](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

8.17 Dokumentacja szablonu struktury testing::internal::EnableIf< bool >

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.18 Dokumentacja struktury testing::internal::EnableIf< true >

```
#include <gtest-internal.h>
```

Typy publiczne

- `typedef void type`

8.18.1 Dokumentacja składowych definicji typu

8.18.1.1 type

```
typedef void testing::internal::EnableIf< true >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.19 Dokumentacja klasy testing::Environment

```
#include <gtest.h>
```

Metody publiczne

- virtual `~Environment ()`
- virtual void `SetUp ()`
- virtual void `TearDown ()`

8.19.1 Dokumentacja konstruktora i destruktora

8.19.1.1 `~Environment()`

```
virtual testing::Environment::~Environment () [inline], [virtual]
```

8.19.2 Dokumentacja funkcji składowych

8.19.2.1 `setUp()`

```
virtual void testing::Environment::setUp () [inline], [virtual]
```

8.19.2.2 `TearDown()`

```
virtual void testing::Environment::TearDown () [inline], [virtual]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

8.20 Dokumentacja szablonu klasy testing::internal::EqHelper< `lhs_is_null_literal` >

```
#include <gtest.h>
```

Statyczne metody publiczne

- template<typename T1, typename T2>
static AssertionResult `Compare` (const char *lhs_expression, const char *rhs_expression, const T1 &lhs, const T2 &rhs)
- static AssertionResult `Compare` (const char *lhs_expression, const char *rhs_expression, `BiggestInt` lhs, `BiggestInt` rhs)

8.20.1 Dokumentacja funkcji składowych

8.20.1.1 Compare() [1/2]

```
template<bool lhs_is_null_literal>
AssertionResult testing::internal::EqHelper< lhs_is_null_literal >::Compare (
    const char * lhs_expression,
    const char * rhs_expression,
    BiggestInt lhs,
    BiggestInt rhs) [inline], [static]
```

8.20.1.2 Compare() [2/2]

```
template<bool lhs_is_null_literal>
template<typename T1, typename T2>
AssertionResult testing::internal::EqHelper< lhs_is_null_literal >::Compare (
    const char * lhs_expression,
    const char * rhs_expression,
    const T1 & lhs,
    const T2 & rhs) [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

8.21 Dokumentacja klasy testing::internal::EqHelper< true >

```
#include <gtest.h>
```

Statyczne metody publiczne

- template<typename T1, typename T2>
 static AssertionResult **Compare** (const char *lhs_expression, const char *rhs_expression, const T1 &lhs, const T2 &rhs, typename **EnableIf<!is_pointer< T2 >::value >::type** *=0)
- template<typename T>
 static AssertionResult **Compare** (const char *lhs_expression, const char *rhs_expression, Secret *, T *rhs)

8.21.1 Dokumentacja funkcji składowych

8.21.1.1 Compare() [1/2]

```
template<typename T1, typename T2>
AssertionResult testing::internal::EqHelper< true >::Compare (
    const char * lhs_expression,
    const char * rhs_expression,
    const T1 & lhs,
    const T2 & rhs,
    typename EnableIf<!is_pointer< T2 >::value >::type * = 0) [inline], [static]
```

8.21.1.2 Compare() [2/2]

```
template<typename T>
AssertionResult testing::internal::EqHelper< true >::Compare (
    const char * lhs_expression,
    const char * rhs_expression,
    Secret * ,
    T * rhs)  [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

8.22 Dokumentacja szablonu klasy testing::internal::FloatingPoint< RawType >

```
#include <gtest-internal.h>
```

Typy publiczne

- typedef TypeWithSize< sizeof(RawType)>::UInt Bits

Metody publiczne

- FloatingPoint (const RawType &x)
- const Bits & bits () const
- Bits exponent_bits () const
- Bits fraction_bits () const
- Bits sign_bit () const
- bool is_nan () const
- bool AlmostEquals (const FloatingPoint &rhs) const
- float Max ()
- double Max ()

Statyczne metody publiczne

- static RawType ReinterpretBits (const Bits bits)
- static RawType Infinity ()
- static RawType Max ()

Statyczne atrybuty publiczne

- static const size_t kBitCount = 8*sizeof(RawType)
- static const size_t kFractionBitCount
- static const size_t kExponentBitCount = kBitCount - 1 - kFractionBitCount
- static const Bits kSignBitMask = static_cast<Bits>(1) << (kBitCount - 1)
- static const Bits kFractionBitMask
- static const Bits kExponentBitMask = ~(kSignBitMask | kFractionBitMask)
- static const size_t kMaxUlps = 4

8.22.1 Dokumentacja składowych definicji typu

8.22.1.1 Bits

```
template<typename RawType>
typedef TypeWithSize<sizeof(RawType)>::UInt testing::internal::FloatingPoint< RawType >::Bits
```

8.22.2 Dokumentacja konstruktora i destruktora

8.22.2.1 FloatingPoint()

```
template<typename RawType>
testing::internal::FloatingPoint< RawType >::FloatingPoint (
    const RawType & x) [inline], [explicit]
```

8.22.3 Dokumentacja funkcji składowych

8.22.3.1 AlmostEquals()

```
template<typename RawType>
bool testing::internal::FloatingPoint< RawType >::AlmostEquals (
    const FloatingPoint< RawType > & rhs) const [inline]
```

8.22.3.2 bits()

```
template<typename RawType>
const Bits & testing::internal::FloatingPoint< RawType >::bits () const [inline]
```

8.22.3.3 exponent_bits()

```
template<typename RawType>
Bits testing::internal::FloatingPoint< RawType >::exponent_bits () const [inline]
```

8.22.3.4 fraction_bits()

```
template<typename RawType>
Bits testing::internal::FloatingPoint< RawType >::fraction_bits () const [inline]
```

8.22.3.5 Infinity()

```
template<typename RawType>
RawType testing::internal::FloatingPoint< RawType >::Infinity () [inline], [static]
```

8.22.3.6 is_nan()

```
template<typename RawType>
bool testing::internal::FloatingPoint< RawType >::is_nan () const [inline]
```

8.22.3.7 Max() [1/3]

```
double testing::internal::FloatingPoint< double >::Max () [inline]
```

8.22.3.8 Max() [2/3]

```
float testing::internal::FloatingPoint< float >::Max () [inline]
```

8.22.3.9 Max() [3/3]

```
template<typename RawType>
RawType testing::internal::FloatingPoint< RawType >::Max () [static]
```

8.22.3.10 ReinterpretBits()

```
template<typename RawType>
RawType testing::internal::FloatingPoint< RawType >::ReinterpretBits (
    const Bits bits) [inline], [static]
```

8.22.3.11 sign_bit()

```
template<typename RawType>
Bits testing::internal::FloatingPoint< RawType >::sign_bit () const [inline]
```

8.22.4 Dokumentacja atrybutów składowych**8.22.4.1 kBitCount**

```
template<typename RawType>
const size_t testing::internal::FloatingPoint< RawType >::kBitCount = 8*sizeof(RawType) [static]
```

8.22.4.2 kExponentBitCount

```
template<typename RawType>
const size_t testing::internal::FloatingPoint< RawType >::kExponentBitCount = kBitCount - 1 -
kFractionBitCount [static]
```

8.22.4.3 kExponentBitMask

```
template<typename RawType>
const Bits testing::internal::FloatingPoint< RawType >::kExponentBitMask = ~(kSignBitMask |
kFractionBitMask) [static]
```

8.22.4.4 kFractionBitCount

```
template<typename RawType>
const size_t testing::internal::FloatingPoint< RawType >::kFractionBitCount [static]
```

Wartość początkowa:

```
=
std::numeric_limits<RawType>::digits - 1
```

8.22.4.5 kFractionBitMask

```
template<typename RawType>
const Bits testing::internal::FloatingPoint< RawType >::kFractionBitMask [static]
```

Wartość początkowa:

```
=
~static_cast<Bits>(0) >> (kExponentBitCount + 1)
```

8.22.4.6 kMaxUlps

```
template<typename RawType>
const size_t testing::internal::FloatingPoint< RawType >::kMaxUlps = 4 [static]
```

8.22.4.7 kSignBitMask

```
template<typename RawType>
const Bits testing::internal::FloatingPoint< RawType >::kSignBitMask = static_cast<Bits>(1)
<< (kBitCount - 1) [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.23 Dokumentacja szablonu klasy testing::internal::FormatForComparison< ToPrint, OtherOperand >

```
#include <gtest-printers.h>
```

Statyczne metody publiczne

- ::std::string Format (const ToPrint &value)

8.23.1 Dokumentacja funkcji składowych

8.23.1.1 Format()

```
template<typename ToPrint, typename OtherOperand>
::std::string testing::internal::FormatForComparison< ToPrint, OtherOperand >::Format (
    const ToPrint & value) [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

8.24 Dokumentacja szablonu klasy testing::internal::FormatForComparison< ToPrint[N], OtherOperand >

```
#include <gtest-printers.h>
```

Statyczne metody publiczne

- ::std::string Format (const ToPrint *value)

8.24.1 Dokumentacja funkcji składowych

8.24.1.1 Format()

```
template<typename ToPrint, size_t N, typename OtherOperand>
::std::string testing::internal::FormatForComparison< ToPrint[N], OtherOperand >::Format (
    const ToPrint * value) [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

8.25 Dokumentacja szablonu klasy std::tr1::gtest_internal::Get< k >

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.26 Dokumentacja klasy std::tr1::gtest_internal::Get< 0 >

```
#include <gtest-tuple.h>
```

Statyczne metody publiczne

- template<class Tuple>
static **GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(0, Tuple))** Field(Tuple &t)
- template<class Tuple>
static **GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(0, Tuple))** ConstField(const Tuple &t)

8.26.1 Dokumentacja funkcji składowych

8.26.1.1 GTEST_ADD_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 0 >:::GTEST_ADD_REF_ (
    GTEST_TUPLE_ELEMENT_(0, Tuple) ) & [inline], [static]
```

8.26.1.2 GTEST_BY_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 0 >:::GTEST_BY_REF_ (
    GTEST_TUPLE_ELEMENT_(0, Tuple) ) const & [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/**gtest-tuple.h**

8.27 Dokumentacja klasy std::tr1::gtest_internal::Get< 1 >

```
#include <gtest-tuple.h>
```

Statyczne metody publiczne

- template<class Tuple>
static **GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(1, Tuple))** Field(Tuple &t)
- template<class Tuple>
static **GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(1, Tuple))** ConstField(const Tuple &t)

8.27.1 Dokumentacja funkcji składowych

8.27.1.1 GTEST_ADD_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 1 >:::GTEST_ADD_REF_ (
    GTEST_TUPLE_ELEMENT_(1, Tuple) ) & [inline], [static]
```

8.27.1.2 GTEST_BY_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 1 >::GTEST_BY_REF_ (
    GTEST_TUPLE_ELEMENT_(1, Tuple) ) const & [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.28 Dokumentacja klasy std::tr1::gtest_internal::Get< 2 >

```
#include <gtest-tuple.h>
```

Statyczne metody publiczne

- template<class Tuple>
 static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(2, Tuple)) Field(Tuple &t)
- template<class Tuple>
 static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(2, Tuple)) ConstField(const Tuple &t)

8.28.1 Dokumentacja funkcji składowych

8.28.1.1 GTEST_ADD_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 2 >::GTEST_ADD_REF_ (
    GTEST_TUPLE_ELEMENT_(2, Tuple) ) & [inline], [static]
```

8.28.1.2 GTEST_BY_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 2 >::GTEST_BY_REF_ (
    GTEST_TUPLE_ELEMENT_(2, Tuple) ) const & [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.29 Dokumentacja klasy std::tr1::gtest_internal::Get< 3 >

```
#include <gtest-tuple.h>
```

Statyczne metody publiczne

- template<class Tuple>
static **GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(3, Tuple))** Field(Tuple &t)
- template<class Tuple>
static **GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(3, Tuple))** ConstField(const Tuple &t)

8.29.1 Dokumentacja funkcji składowych

8.29.1.1 GTEST_ADD_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 3 >::GTEST_ADD_REF_(
    GTEST_TUPLE_ELEMENT_(3, Tuple) ) & [inline], [static]
```

8.29.1.2 GTEST_BY_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 3 >::GTEST_BY_REF_(
    GTEST_TUPLE_ELEMENT_(3, Tuple) ) const & [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/**gtest-tuple.h**

8.30 Dokumentacja klasy std::tr1::gtest_internal::Get< 4 >

```
#include <gtest-tuple.h>
```

Statyczne metody publiczne

- template<class Tuple>
static **GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(4, Tuple))** Field(Tuple &t)
- template<class Tuple>
static **GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(4, Tuple))** ConstField(const Tuple &t)

8.30.1 Dokumentacja funkcji składowych

8.30.1.1 GTEST_ADD_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 4 >::GTEST_ADD_REF_(
    GTEST_TUPLE_ELEMENT_(4, Tuple) ) & [inline], [static]
```

8.30.1.2 GTEST_BY_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 4 >::GTEST_BY_REF_ (
    GTEST_TUPLE_ELEMENT_(4, Tuple) ) const & [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.31 Dokumentacja klasy std::tr1::gtest_internal::Get< 5 >

```
#include <gtest-tuple.h>
```

Statyczne metody publiczne

- template<class Tuple>
 static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(5, Tuple)) Field(Tuple &t)
- template<class Tuple>
 static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(5, Tuple)) ConstField(const Tuple &t)

8.31.1 Dokumentacja funkcji składowych

8.31.1.1 GTEST_ADD_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 5 >::GTEST_ADD_REF_ (
    GTEST_TUPLE_ELEMENT_(5, Tuple) ) & [inline], [static]
```

8.31.1.2 GTEST_BY_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 5 >::GTEST_BY_REF_ (
    GTEST_TUPLE_ELEMENT_(5, Tuple) ) const & [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.32 Dokumentacja klasy std::tr1::gtest_internal::Get< 6 >

```
#include <gtest-tuple.h>
```

Statyczne metody publiczne

- template<class Tuple>
static **GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(6, Tuple))** Field(Tuple &t)
- template<class Tuple>
static **GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(6, Tuple))** ConstField(const Tuple &t)

8.32.1 Dokumentacja funkcji składowych

8.32.1.1 GTEST_ADD_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 6 >::GTEST_ADD_REF_(
    GTEST_TUPLE_ELEMENT_(6, Tuple) ) & [inline], [static]
```

8.32.1.2 GTEST_BY_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 6 >::GTEST_BY_REF_(
    GTEST_TUPLE_ELEMENT_(6, Tuple) ) const & [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/**gtest-tuple.h**

8.33 Dokumentacja klasy std::tr1::gtest_internal::Get< 7 >

```
#include <gtest-tuple.h>
```

Statyczne metody publiczne

- template<class Tuple>
static **GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(7, Tuple))** Field(Tuple &t)
- template<class Tuple>
static **GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(7, Tuple))** ConstField(const Tuple &t)

8.33.1 Dokumentacja funkcji składowych

8.33.1.1 GTEST_ADD_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 7 >::GTEST_ADD_REF_(
    GTEST_TUPLE_ELEMENT_(7, Tuple) ) & [inline], [static]
```

8.33.1.2 GTEST_BY_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 7 >::GTEST_BY_REF_ (
    GTEST_TUPLE_ELEMENT_(7, Tuple) ) const & [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.34 Dokumentacja klasy std::tr1::gtest_internal::Get< 8 >

```
#include <gtest-tuple.h>
```

Statyczne metody publiczne

- template<class Tuple>
 static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(8, Tuple)) Field(Tuple &t)
- template<class Tuple>
 static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(8, Tuple)) ConstField(const Tuple &t)

8.34.1 Dokumentacja funkcji składowych

8.34.1.1 GTEST_ADD_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 8 >::GTEST_ADD_REF_ (
    GTEST_TUPLE_ELEMENT_(8, Tuple) ) & [inline], [static]
```

8.34.1.2 GTEST_BY_REF_()

```
template<class Tuple>
std::tr1::gtest_internal::Get< 8 >::GTEST_BY_REF_ (
    GTEST_TUPLE_ELEMENT_(8, Tuple) ) const & [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.35 Dokumentacja klasy std::tr1::gtest_internal::Get< 9 >

```
#include <gtest-tuple.h>
```

Statyczne metody publiczne

- template<class Tuple>
static [GTEST_ADD_REF_\(GTEST_TUPLE_ELEMENT_\(9, Tuple\)\)](#) Field(Tuple &t)
- template<class Tuple>
static [GTEST_BY_REF_\(GTEST_TUPLE_ELEMENT_\(9, Tuple\)\)](#) ConstField(const Tuple &t)

8.35.1 Dokumentacja funkcji składowych

8.35.1.1 GTEST_ADD_REF_()

```
template<class Tuple>
std::tr1::gtest\_internal::Get< 9 >::GTEST\_ADD\_REF\_ (
    GTEST\_TUPLE\_ELEMENT\_\(9, Tuple\)  & [inline], [static]
```

8.35.1.2 GTEST_BY_REF_()

```
template<class Tuple>
std::tr1::gtest\_internal::Get< 9 >::GTEST\_BY\_REF\_ (
    GTEST\_TUPLE\_ELEMENT\_\(9, Tuple\)  const & [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-tuple.h](#)

8.36 Dokumentacja klasy testing::internal::GTestLog

```
#include <gtest-port.h>
```

Metody publiczne

- [GTestLog \(GTestLogSeverity severity, const char *file, int line\)](#)
- [~GTestLog \(\)](#)
- [::std::ostream & GetStream \(\)](#)

8.36.1 Dokumentacja konstruktora i destruktora

8.36.1.1 GTestLog()

```
testing::internal::GTestLog::GTestLog (
    GTestLogSeverity severity,
    const char \* file,
    int line\)
```

8.36.1.2 ~GTestLog()

```
testing::internal::GTestLog::~GTestLog ()
```

8.36.2 Dokumentacja funkcji składowych

8.36.2.1 GetStream()

```
::std::ostream & testing::internal::GTestLog::GetStream () [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.37 Dokumentacja klasy testing::internal::GTestMutexLock

```
#include <gtest-port.h>
```

Metody publiczne

- [GTestMutexLock \(Mutex *\)](#)

8.37.1 Dokumentacja konstruktora i destruktora

8.37.1.1 GTestMutexLock()

```
testing::internal::GTestMutexLock::GTestMutexLock (
    Mutex * ) [inline], [explicit]
```

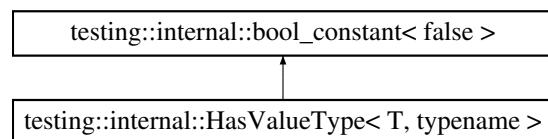
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.38 Dokumentacja szablonu struktury testing::internal::HasValueType< T, typename >

```
#include <gtest-internal.h>
```

Diagram dziedziczenia dla testing::internal::HasValueType< T, typename >



Dodatkowe dziedziczone składowe

Typy publiczne dziedziczone z testing::internal::bool_constant< false >

- `typedef bool_constant< bool_value > type`

Statyczne atrybuty publiczne dziedziczone z testing::internal::bool_constant< false >

- `static const bool value`

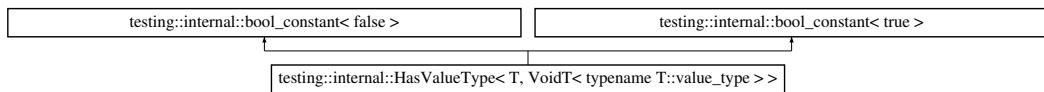
Dokumentacja dla tej struktury została wygenerowana z pliku:

- `packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h`

8.39 Dokumentacja szablonu struktury testing::internal::HasValueType< T, VoidT< typename T::value_type > >

```
#include <gtest-internal.h>
```

Diagram dziedziczenia dla testing::internal::HasValueType< T, VoidT< typename T::value_type > >



Dodatkowe dziedziczone składowe

Typy publiczne dziedziczone z testing::internal::bool_constant< false >

- `typedef bool_constant< bool_value > type`

Typy publiczne dziedziczone z testing::internal::bool_constant< true >

- `typedef bool_constant< bool_value > type`

Statyczne atrybuty publiczne dziedziczone z testing::internal::bool_constant< false >

- `static const bool value`

Statyczne atrybuty publiczne dziedziczone z testing::internal::bool_constant< true >

- `static const bool value`

Dokumentacja dla tej struktury została wygenerowana z pliku:

- `packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h`

8.40 Dokumentacja szablonu klasy testing::internal::ImplicitlyConvertible< From, To >

```
#include <gtest-internal.h>
```

Statyczne atrybuty publiczne

- static const bool `value`

8.40.1 Dokumentacja atrybutów składowych

8.40.1.1 `value`

```
template<typename From, typename To>
const bool testing::internal::ImplicitlyConvertible< From, To >::value [static]
```

Wartość początkowa:

```
=
sizeof(Helper(ImplicitlyConvertible::MakeFrom())) == 1
```

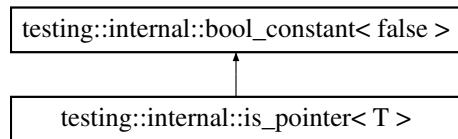
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.41 Dokumentacja szablonu struktury testing::internal::is_pointer< T >

```
#include <gtest-port.h>
```

Diagram dziedziczenia dla `testing::internal::is_pointer< T >`



Dodatkowe dziedziczone składowe

Typy publiczne dziedziczone z `testing::internal::bool_constant< false >`

- typedef `bool_constant< bool_value > type`

Statyczne atrybuty publiczne dziedziczone z testing::internal::bool_constant< false >

- static const bool [value](#)

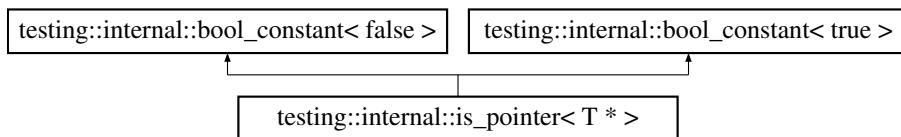
Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.42 Dokumentacja szablonu struktury testing::internal::is_pointer< T * >

```
#include <gtest-port.h>
```

Diagram dziedziczenia dla testing::internal::is_pointer< T * >



Dodatkowe dziedziczone składowe

Typy publiczne dziedziczone z testing::internal::bool_constant< false >

- `typedef bool_constant< bool_value > type`

Typy publiczne dziedziczone z testing::internal::bool_constant< true >

- `typedef bool_constant< bool_value > type`

Statyczne atrybuty publiczne dziedziczone z testing::internal::bool_constant< false >

- static const bool [value](#)

Statyczne atrybuty publiczne dziedziczone z testing::internal::bool_constant< true >

- static const bool [value](#)

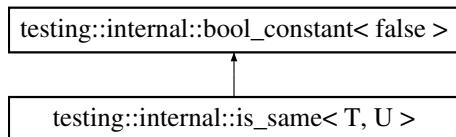
Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.43 Dokumentacja szablonu struktury testing::internal::is_same< T, U >

```
#include <gtest-port.h>
```

Diagram dziedziczenia dla testing::internal::is_same< T, U >



Dodatkowe dziedziczone składowe

Typy publiczne dziedziczone z [testing::internal::bool_constant< false >](#)

- [typedef bool_constant< bool_value > type](#)

Statyczne atrybuty publiczne dziedziczone z [testing::internal::bool_constant< false >](#)

- [static const bool value](#)

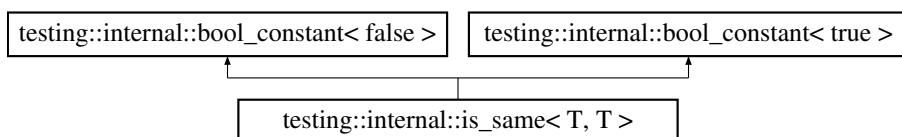
Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.44 Dokumentacja szablonu struktury testing::internal::is_same< T, T >

```
#include <gtest-port.h>
```

Diagram dziedziczenia dla testing::internal::is_same< T, T >



Dodatkowe dziedziczone składowe

Typy publiczne dziedziczone z [testing::internal::bool_constant< false >](#)

- [typedef bool_constant< bool_value > type](#)

Typy publiczne dziedziczone z testing::internal::bool_constant< true >

- typedef bool_constant< bool_value > type

Statyczne atrybuty publiczne dziedziczone z testing::internal::bool_constant< false >

- static const bool value

Statyczne atrybuty publiczne dziedziczone z testing::internal::bool_constant< true >

- static const bool value

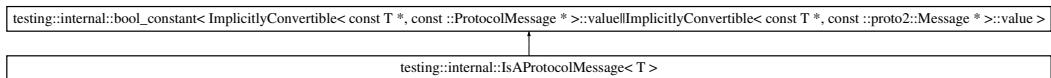
Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.45 Dokumentacja szablonu struktury testing::internal::IsAProtocolMessage< T >

```
#include <gtest-internal.h>
```

Diagram dziedziczenia dla testing::internal::IsAProtocolMessage< T >

**Dodatkowe dziedziczone składowe****Typy publiczne dziedziczone z****testing::internal::bool_constant< ImplicitlyConvertible< const T *, const ::ProtocolMessage * >::value || ImplicitlyConvertible< const T *, const ::proto2::Message * >::value >**

- typedef bool_constant< bool_value > type

Statyczne atrybuty publiczne dziedziczone z**testing::internal::bool_constant< ImplicitlyConvertible< const T *, const ::ProtocolMessage * >::value || ImplicitlyConvertible< const T *, const ::proto2::Message * >::value >**

- static const bool value

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.46 Dokumentacja szablonu struktury `testing::internal::IsHashTable< T >`

```
#include <gtest-internal.h>
```

Statyczne atrybuty publiczne

- static const bool `value` = sizeof(test<T>(0, 0)) == sizeof(int)

8.46.1 Dokumentacja atrybutów składowych

8.46.1.1 `value`

```
template<typename T>
const bool testing::internal::IsHashTable< T >::value = sizeof(test<T>(0, 0)) == sizeof(int)
[static]
```

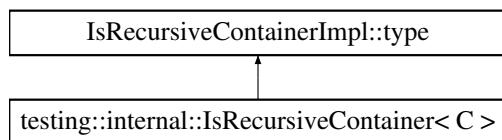
Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.47 Dokumentacja szablonu struktury `testing::internal::IsRecursiveContainer< C >`

```
#include <gtest-internal.h>
```

Diagram dziedziczenia dla `testing::internal::IsRecursiveContainer< C >`



Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.48 Dokumentacja szablonu struktury `testing::internal::IsRecursiveContainerImpl< C, bool, bool >`

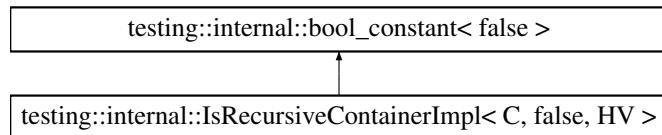
Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.49 Dokumentacja szablonu struktury testing::internal::IsRecursiveContainerImpl< C, false, HV >

```
#include <gtest-internal.h>
```

Diagram dziedziczenia dla testing::internal::IsRecursiveContainerImpl< C, false, HV >



Dodatkowe dziedziczone składowe

Typy publiczne dziedziczone z [testing::internal::bool_constant< false >](#)

- [typedef bool_constant< bool_value > type](#)

Statyczne atrybuty publiczne dziedziczone z [testing::internal::bool_constant< false >](#)

- [static const bool value](#)

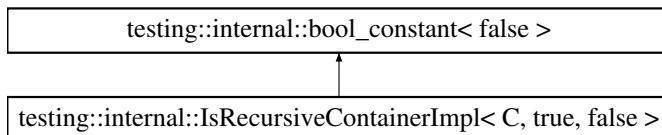
Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.50 Dokumentacja szablonu struktury testing::internal::IsRecursiveContainerImpl< C, true, false >

```
#include <gtest-internal.h>
```

Diagram dziedziczenia dla testing::internal::IsRecursiveContainerImpl< C, true, false >



Dodatkowe dziedziczone składowe

Typy publiczne dziedziczone z [testing::internal::bool_constant< false >](#)

- [typedef bool_constant< bool_value > type](#)

Statyczne atrybuty publiczne dziedziczone z `testing::internal::bool_constant< false >`

- static const bool `value`

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.51 Dokumentacja szablonu struktury

`testing::internal::IsRecursiveContainerImpl< C, true, true >`

```
#include <gtest-internal.h>
```

Typy publiczne

- typedef `IteratorTraits< typenameC::iterator >::value_type value_type`
- typedef `is_same< value_type, C > type`

8.51.1 Dokumentacja składowych definicji typu

8.51.1.1 type

```
template<typename C>
typedef is_same<value_type, C> testing::internal::IsRecursiveContainerImpl< C, true, true >::type
```

8.51.1.2 value_type

```
template<typename C>
typedef IteratorTraits<typenameC::iterator>::value_type testing::internal::IsRecursiveContainerImpl< C, true, true >::value_type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.52 Dokumentacja szablonu struktury `testing::internal::IsSame< T, U >`

```
#include <gtest-port.h>
```

Typy publiczne

- enum { `value` = false }

8.52.1 Dokumentacja składowych wyliczanych

8.52.1.1 anonymous enum

```
template<typename T, typename U>
anonymous enum
```

Wartości wyliczeń

value	
-------	--

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.53 Dokumentacja szablonu struktury testing::internal::IsSame< T, T >

```
#include <gtest-port.h>
```

Typy publiczne

- enum { **value** = true }
- enum

8.53.1 Dokumentacja składowych wyliczanych

8.53.1.1 anonymous enum

```
template<typename T>
anonymous enum
```

Wartości wyliczeń

value	
-------	--

8.53.1.2 anonymous enum

```
anonymous enum
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.54 Dokumentacja szablonu struktury testing::internal::IteratorTraits< Iterator >

```
#include <gtest-port.h>
```

Typy publiczne

- typedef Iterator::value_type **value_type**

8.54.1 Dokumentacja składowych definicji typu

8.54.1.1 value_type

```
template<typename Iterator>
typedef Iterator::value_type testing::internal::IteratorTraits< Iterator >::value_type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.55 Dokumentacja szablonu struktury testing::internal::IteratorTraits< const T * >

```
#include <gtest-port.h>
```

Typy publiczne

- typedef T value_type

8.55.1 Dokumentacja składowych definicji typu

8.55.1.1 value_type

```
template<typename T>
typedef T testing::internal::IteratorTraits< const T * >::value_type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.56 Dokumentacja szablonu struktury testing::internal::IteratorTraits< T * >

```
#include <gtest-port.h>
```

Typy publiczne

- typedef T value_type

8.56.1 Dokumentacja składowych definicji typu

8.56.1.1 value_type

```
template<typename T>
typedef T testing::internal::IteratorTraits< T * >::value_type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.57 Dokumentacja szablonu klasy testing::internal::linked_ptr< T >

```
#include <gtest-linked_ptr.h>
```

Typy publiczne

- typedef T element_type

Metody publiczne

- linked_ptr (T *ptr=NULL)
- ~linked_ptr ()
- template<typename U>
linked_ptr (linked_ptr< U > const &ptr)
- linked_ptr (linked_ptr const &ptr)
- template<typename U>
linked_ptr & operator= (linked_ptr< U > const &ptr)
- linked_ptr & operator= (linked_ptr const &ptr)
- void reset (T *ptr=NULL)
- T * get () const
- T * operator-> () const
- T & operator* () const
- bool operator== (T *p) const
- bool operator!= (T *p) const
- template<typename U>
bool operator== (linked_ptr< U > const &ptr) const
- template<typename U>
bool operator!= (linked_ptr< U > const &ptr) const

Przyjaciele

- template<typename U>
class linked_ptr

8.57.1 Dokumentacja składowych definicji typu

8.57.1.1 element_type

```
template<typename T>
typedef T testing::internal::linked_ptr< T >::element_type
```

8.57.2 Dokumentacja konstruktora i destruktora

8.57.2.1 linked_ptr() [1/3]

```
template<typename T>
testing::internal::linked_ptr< T >::linked_ptr (
    T * ptr = NULL)  [inline], [explicit]
```

8.57.2.2 ~linked_ptr()

```
template<typename T>
testing::internal::linked_ptr< T >::~linked_ptr ()  [inline]
```

8.57.2.3 linked_ptr() [2/3]

```
template<typename T>
template<typename U>
testing::internal::linked_ptr< T >::linked_ptr (
    linked_ptr< U > const & ptr)  [inline]
```

8.57.2.4 linked_ptr() [3/3]

```
template<typename T>
testing::internal::linked_ptr< T >::linked_ptr (
    linked_ptr< T > const & ptr)  [inline]
```

8.57.3 Dokumentacja funkcji składowych

8.57.3.1 get()

```
template<typename T>
T * testing::internal::linked_ptr< T >::get () const  [inline]
```

8.57.3.2 operator"!=() [1/2]

```
template<typename T>
template<typename U>
bool testing::internal::linked_ptr< T >::operator!= (
    linked_ptr< U > const & ptr) const  [inline]
```

8.57.3.3 operator"!=() [2/2]

```
template<typename T>
bool testing::internal::linked_ptr< T >::operator!= (
    T * p) const [inline]
```

8.57.3.4 operator*()

```
template<typename T>
T & testing::internal::linked_ptr< T >::operator* () const [inline]
```

8.57.3.5 operator->()

```
template<typename T>
T * testing::internal::linked_ptr< T >::operator-> () const [inline]
```

8.57.3.6 operator=() [1/2]

```
template<typename T>
linked_ptr & testing::internal::linked_ptr< T >::operator= (
    linked_ptr< T > const & ptr) [inline]
```

8.57.3.7 operator=() [2/2]

```
template<typename T>
template<typename U>
linked_ptr & testing::internal::linked_ptr< T >::operator= (
    linked_ptr< U > const & ptr) [inline]
```

8.57.3.8 operator==() [1/2]

```
template<typename T>
template<typename U>
bool testing::internal::linked_ptr< T >::operator== (
    linked_ptr< U > const & ptr) const [inline]
```

8.57.3.9 operator==() [2/2]

```
template<typename T>
bool testing::internal::linked_ptr< T >::operator== (
    T * p) const [inline]
```

8.57.3.10 reset()

```
template<typename T>
void testing::internal::linked_ptr< T >::reset (
    T * ptr = NULL) [inline]
```

8.57.4 Dokumentacja przyjaciół i powiązanych symboli

8.57.4.1 linked_ptr

```
template<typename T>
template<typename U>
friend class linked_ptr [friend]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-linked_ptr](#).

8.58 Dokumentacja klasy testing::internal::linked_ptr_internal

```
#include <gtest-linked_ptr.h>
```

Metody publiczne

- void [join_new](#) ()
- void [join](#) (linked_ptr_internal const *ptr) [GTEST_LOCK_EXCLUDED](#)(g_linked_ptr_mutex)
- bool [depart](#) () [GTEST_LOCK_EXCLUDED](#)(g_linked_ptr_mutex)

8.58.1 Dokumentacja funkcji składowych

8.58.1.1 depart()

```
bool testing::internal::linked_ptr_internal::depart () [inline]
```

8.58.1.2 join()

```
void testing::internal::linked_ptr_internal::join (
    linked_ptr_internal const * ptr) [inline]
```

8.58.1.3 join_new()

```
void testing::internal::linked_ptr_internal::join_new () [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-linked_ptr](#).

8.59 Dokumentacja szablonu klasy MergeSort< T >

Klasa szablonowa realizująca algorytm sortowania przez scalanie.

```
#include <MergeSort.h>
```

Statyczne metody publiczne

- static void [sort](#) (std::vector< T > &arr)

8.59.1 Opis szczegółowy

```
template<typename T>
class MergeSort< T >
```

Klasa szablonowa realizująca algorytm sortowania przez scalanie.

Parametry Szablonu

T	Typ danych (np. int, double, float).
---	--------------------------------------

8.59.2 Dokumentacja funkcji składowych

8.59.2.1 sort()

```
template<typename T>
void MergeSort< T >::sort (
    std::vector< T > & arr) [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- MergeSortApp/[MergeSort.h](#)

8.60 Dokumentacja klasy testing::Message

```
#include <gtest-message.h>
```

Metody publiczne

- [Message](#) ()
- [Message](#) (const Message &msg)
- [Message](#) (const char *str)
- template<typename T>
 [Message](#) & [operator<<](#) (const T &val)
- template<typename T>
 [Message](#) & [operator<<](#) (T *const &pointer)
- [Message](#) & [operator<<](#) (BasicNarrowIoManip val)
- [Message](#) & [operator<<](#) (bool b)
- [Message](#) & [operator<<](#) (const wchar_t *wide_c_str)
- [Message](#) & [operator<<](#) (wchar_t *wide_c_str)
- std::string [GetString](#) () const

8.60.1 Dokumentacja konstruktora i destruktora

8.60.1.1 Message() [1/3]

```
testing:::Message:::Message ()
```

8.60.1.2 Message() [2/3]

```
testing:::Message:::Message (
    const Message & msg) [inline]
```

8.60.1.3 Message() [3/3]

```
testing:::Message:::Message (
    const char * str) [inline], [explicit]
```

8.60.2 Dokumentacja funkcji składowych

8.60.2.1 GetString()

```
std::string testing:::Message:::GetString () const
```

8.60.2.2 operator<<() [1/6]

```
Message & testing:::Message:::operator<< (
    BasicNarrowIoManip val) [inline]
```

8.60.2.3 operator<<() [2/6]

```
Message & testing:::Message:::operator<< (
    bool b) [inline]
```

8.60.2.4 operator<<() [3/6]

```
template<typename T>
Message & testing:::Message:::operator<< (
    const T & val) [inline]
```

8.60.2.5 operator<<() [4/6]

```
Message & testing:::Message:::operator<< (
    const wchar_t * wide_c_str)
```

8.60.2.6 operator<<() [5/6]

```
template<typename T>
Message & testing::Message::operator<< (
    T *const & pointer) [inline]
```

8.60.2.7 operator<<() [6/6]

```
Message & testing::Message::operator<< (
    wchar_t * wide_c_str)
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-message.h

8.61 Dokumentacja klasy testing::internal::Mutex

```
#include <gtest-port.h>
```

Metody publiczne

- [Mutex \(\)](#)
- void [Lock \(\)](#)
- void [Unlock \(\)](#)
- void [AssertHeld \(\) const](#)

8.61.1 Dokumentacja konstruktora i destruktora

8.61.1.1 Mutex()

```
testing::internal::Mutex::Mutex () [inline]
```

8.61.2 Dokumentacja funkcji składowych

8.61.2.1 AssertHeld()

```
void testing::internal::Mutex::AssertHeld () const [inline]
```

8.61.2.2 Lock()

```
void testing::internal::Mutex::Lock () [inline]
```

8.61.2.3 Unlock()

```
void testing::internal::Mutex::Unlock () [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.62 Dokumentacja szablonu klasy testing::internal::NativeArray<Element >

```
#include <gtest-internal.h>
```

Typy publiczne

- `typedef Element value_type`
- `typedef Element * iterator`
- `typedef const Element * const_iterator`

Metody publiczne

- `NativeArray (const Element *array, size_t count, RelationToSourceReference)`
- `NativeArray (const Element *array, size_t count, RelationToSourceCopy)`
- `NativeArray (const NativeArray &rhs)`
- `~NativeArray ()`
- `size_t size () const`
- `const_iterator begin () const`
- `const_iterator end () const`
- `bool operator== (const NativeArray &rhs) const`

8.62.1 Dokumentacja składowych definicji typu

8.62.1.1 const_iterator

```
template<typename Element>
typedef const Element* testing::internal::NativeArray< Element >::const_iterator
```

8.62.1.2 iterator

```
template<typename Element>
typedef Element* testing::internal::NativeArray< Element >::iterator
```

8.62.1.3 value_type

```
template<typename Element>
typedef Element testing::internal::NativeArray< Element >::value_type
```

8.62.2 Dokumentacja konstruktora i destruktora

8.62.2.1 NativeArray() [1/3]

```
template<typename Element>
testing::internal::NativeArray< Element >::NativeArray (
    const Element * array,
    size_t count,
    RelationToSourceReference ) [inline]
```

8.62.2.2 NativeArray() [2/3]

```
template<typename Element>
testing::internal::NativeArray< Element >::NativeArray (
    const Element * array,
    size_t count,
    RelationToSourceCopy ) [inline]
```

8.62.2.3 NativeArray() [3/3]

```
template<typename Element>
testing::internal::NativeArray< Element >::NativeArray (
    const NativeArray< Element > & rhs) [inline]
```

8.62.2.4 ~NativeArray()

```
template<typename Element>
testing::internal::NativeArray< Element >::~NativeArray () [inline]
```

8.62.3 Dokumentacja funkcji składowych

8.62.3.1 begin()

```
template<typename Element>
const_iterator testing::internal::NativeArray< Element >::begin () const [inline]
```

8.62.3.2 end()

```
template<typename Element>
const_iterator testing::internal::NativeArray< Element >::end () const [inline]
```

8.62.3.3 operator==()

```
template<typename Element>
bool testing::internal::NativeArray< Element >::operator== (
    const NativeArray< Element > & rhs) const [inline]
```

8.62.3.4 size()

```
template<typename Element>
size_t testing::internal::NativeArray< Element >::size () const [inline]
```

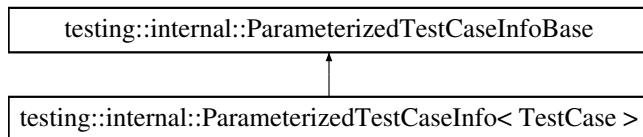
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.63 Dokumentacja szablonu klasy testing::internal::ParameterizedTestCaseInfo< TestCase >

```
#include <gtest-param-util.h>
```

Diagram dziedziczenia dla testing::internal::ParameterizedTestCaseInfo< TestCase >



Typy publiczne

- `typedef TestCase::ParamType ParamType`
- `typedef ParamNameGenFunc< ParamType >::Type ParamNameGeneratorFunc`

Metody publiczne

- `typedef ParamGenerator (GeneratorCreationFunc())`
- `ParameterizedTestCaseInfo (const char *name, CodeLocation code_location)`
- `virtual const std::string & GetTestCaseName () const`
- `virtual Typeld GetTestCaseTypeld () const`
- `void AddTestPattern (const char *test_case_name, const char *test_base_name, TestMetaFactoryBase< ParamType > *meta_factory)`
- `int AddTestCaseInstantiation (const std::string &instantiation_name, GeneratorCreationFunc *func, ParamNameGeneratorFunc *name_func, const char *file, int line)`
- `virtual void RegisterTests ()`

Metody publiczne dziedziczone z testing::internal::ParameterizedTestCaseInfoBase

- `virtual ~ParameterizedTestCaseInfoBase ()`

Dodatkowe dziedziczone składowe

Metody chronione dziedziczone z testing::internal::ParameterizedTestCaseInfoBase

- ParameterizedTestCaseInfoBase ()

8.63.1 Dokumentacja składowych definicji typu

8.63.1.1 ParamNameGeneratorFunc

```
template<class TestCase>
typedef ParamNameGenFunc<ParamType>::Type testing::internal::ParameterizedTestCaseInfo< TestCase >::ParamNameGeneratorFunc
```

8.63.1.2 ParamType

```
template<class TestCase>
typedef TestCase::ParamType testing::internal::ParameterizedTestCaseInfo< TestCase >::ParamType
```

8.63.2 Dokumentacja konstruktora i destruktora

8.63.2.1 ParameterizedTestCaseInfo()

```
template<class TestCase>
testing::internal::ParameterizedTestCaseInfo< TestCase >::ParameterizedTestCaseInfo (
    const char * name,
    CodeLocation code_location) [inline], [explicit]
```

8.63.3 Dokumentacja funkcji składowych

8.63.3.1 AddTestCaseInstantiation()

```
template<class TestCase>
int testing::internal::ParameterizedTestCaseInfo< TestCase >::AddTestCaseInstantiation (
    const std::string & instantiation_name,
    GeneratorCreationFunc * func,
    ParamNameGeneratorFunc * name_func,
    const char * file,
    int line) [inline]
```

8.63.3.2 AddTestPattern()

```
template<class TestCase>
void testing::internal::ParameterizedTestCaseInfo< TestCase >::AddTestPattern (
    const char * test_case_name,
    const char * test_base_name,
    TestMetaFactoryBase< ParamType > * meta_factory) [inline]
```

8.63.3.3 GetTestCaseName()

```
template<class TestCase>
virtual const std::string & testing::internal::ParameterizedTestCaseInfo< TestCase >::Get<-
TestCaseName () const [inline], [virtual]
```

Implementuje [testing::internal::ParameterizedTestCaseInfoBase](#).

8.63.3.4 GetTestCaseTypeId()

```
template<class TestCase>
virtual TypeId testing::internal::ParameterizedTestCaseInfo< TestCase >::GetTestCaseTypeId () const [inline], [virtual]
```

Implementuje [testing::internal::ParameterizedTestCaseInfoBase](#).

8.63.3.5 ParamGenerator()

```
template<class TestCase>
typedef testing::internal::ParameterizedTestCaseInfo< TestCase >::ParamGenerator ( GeneratorCreationFunc )
```

8.63.3.6 RegisterTests()

```
template<class TestCase>
virtual void testing::internal::ParameterizedTestCaseInfo< TestCase >::RegisterTests () [inline], [virtual]
```

Implementuje [testing::internal::ParameterizedTestCaseInfoBase](#).

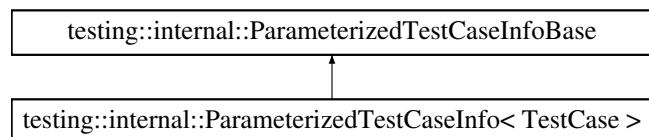
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.64 Dokumentacja klasy testing::internal::ParameterizedTestCaseInfoBase

```
#include <gtest-param-util.h>
```

Diagram dziedziczenia dla [testing::internal::ParameterizedTestCaseInfoBase](#)



Metody publiczne

- virtual ~ParameterizedTestCaseInfoBase ()
- virtual const std::string & GetTestCaseName () const =0
- virtual Typeld GetTestCaseTypeId () const =0
- virtual void RegisterTests ()=0

Metody chronione

- ParameterizedTestCaseInfoBase ()

8.64.1 Dokumentacja konstruktora i destruktora

8.64.1.1 ~ParameterizedTestCaseInfoBase()

```
virtual testing::internal::ParameterizedTestCaseInfoBase::~ParameterizedTestCaseInfoBase ()  
[inline], [virtual]
```

8.64.1.2 ParameterizedTestCaseInfoBase()

```
testing::internal::ParameterizedTestCaseInfoBase::ParameterizedTestCaseInfoBase () [inline],  
[protected]
```

8.64.2 Dokumentacja funkcji składowych

8.64.2.1 GetTestCaseName()

```
virtual const std::string & testing::internal::ParameterizedTestCaseInfoBase::GetTestCaseName  
() const [pure virtual]
```

Implementowany w [testing::internal::ParameterizedTestCaseInfo< TestCase >](#).

8.64.2.2 GetTestCaseTypeId()

```
virtual Typeld testing::internal::ParameterizedTestCaseInfoBase::GetTestCaseTypeId () const  
[pure virtual]
```

Implementowany w [testing::internal::ParameterizedTestCaseInfo< TestCase >](#).

8.64.2.3 RegisterTests()

```
virtual void testing::internal::ParameterizedTestCaseInfoBase::RegisterTests () [pure virtual]
```

Implementowany w [testing::internal::ParameterizedTestCaseInfo< TestCase >](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.65 Dokumentacja klasy testing::internal::ParameterizedTestCaseRegistry

```
#include <gtest-param-util.h>
```

Metody publiczne

- ParameterizedTestCaseRegistry ()
- ~ParameterizedTestCaseRegistry ()
- template<class TestCase>
ParameterizedTestCaseInfo< TestCase > * GetTestCasePatternHolder (const char *test_case_name,
CodeLocation code_location)
- void RegisterTests ()

8.65.1 Dokumentacja konstruktora i destruktora

8.65.1.1 ParameterizedTestCaseRegistry()

```
testing::internal::ParameterizedTestCaseRegistry::ParameterizedTestCaseRegistry () [inline]
```

8.65.1.2 ~ParameterizedTestCaseRegistry()

```
testing::internal::ParameterizedTestCaseRegistry::~ParameterizedTestCaseRegistry () [inline]
```

8.65.2 Dokumentacja funkcji składowych

8.65.2.1 GetTestCasePatternHolder()

```
template<class TestCase>
ParameterizedTestCaseInfo< TestCase > * testing::internal::ParameterizedTestCaseRegistry::GetTestCasePatternHolder (
    const char * test_case_name,
    CodeLocation code_location) [inline]
```

8.65.2.2 RegisterTests()

```
void testing::internal::ParameterizedTestCaseRegistry::RegisterTests () [inline]
```

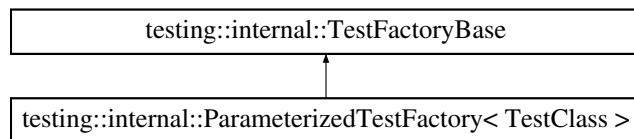
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.66 Dokumentacja szablonu klasy testing::internal::ParameterizedTestFactory< TestClass >

```
#include <gtest-param-util.h>
```

Diagram dziedziczenia dla testing::internal::ParameterizedTestFactory< TestClass >



Typy publiczne

- `typedef TestClass::ParamType ParamType`

Metody publiczne

- `ParameterizedTestFactory (ParamType parameter)`
- `virtual Test * CreateTest ()`

Metody publiczne dziedziczone z testing::internal::TestFactoryBase

- `virtual ~TestFactoryBase ()`

Dodatkowe dziedziczone składowe

Metody chronione dziedziczone z testing::internal::TestFactoryBase

- `TestFactoryBase ()`

8.66.1 Dokumentacja składowych definicji typu

8.66.1.1 ParamType

```
template<class TestClass>
typedef TestClass::ParamType testing::internal::ParameterizedTestFactory< TestClass >::ParamType
```

8.66.2 Dokumentacja konstruktora i destruktora

8.66.2.1 ParameterizedTestFactory()

```
template<class TestClass>
testing::internal::ParameterizedTestFactory< TestClass >::ParameterizedTestFactory (
    ParamType parameter) [inline], [explicit]
```

8.66.3 Dokumentacja funkcji składowych

8.66.3.1 CreateTest()

```
template<class TestClass>
virtual Test * testing::internal::ParameterizedTestFactory< TestClass >::CreateTest () [inline],
[virtual]
```

Implementuje [testing::internal::TestFactoryBase](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-param-util.h](#)

8.67 Dokumentacja szablonu klasy [testing::internal::ParamGenerator](#)< T >

```
#include <gtest-param-util.h>
```

Typy publiczne

- [typedef ParamIterator](#)< T > iterator

Metody publiczne

- [ParamGenerator](#) ([ParamGeneratorInterface](#)< T > *impl)
- [ParamGenerator](#) (const [ParamGenerator](#) &other)
- [ParamGenerator](#) & [operator=](#) (const [ParamGenerator](#) &other)
- [iterator begin](#) () const
- [iterator end](#) () const

8.67.1 Dokumentacja składowych definicji typu

8.67.1.1 iterator

```
template<typename T>
typedef ParamIterator<T> testing::internal::ParamGenerator< T >::iterator
```

8.67.2 Dokumentacja konstruktora i destruktora

8.67.2.1 [ParamGenerator\(\)](#) [1/2]

```
template<typename T>
testing::internal::ParamGenerator< T >::ParamGenerator (
    ParamGeneratorInterface< T > * impl) [inline], [explicit]
```

8.67.2.2 ParamGenerator() [2/2]

```
template<typename T>
testing::internal::ParamGenerator< T >::ParamGenerator (
    const ParamGenerator< T > & other) [inline]
```

8.67.3 Dokumentacja funkcji składowych

8.67.3.1 begin()

```
template<typename T>
iterator testing::internal::ParamGenerator< T >::begin () const [inline]
```

8.67.3.2 end()

```
template<typename T>
iterator testing::internal::ParamGenerator< T >::end () const [inline]
```

8.67.3.3 operator=()

```
template<typename T>
ParamGenerator & testing::internal::ParamGenerator< T >::operator= (
    const ParamGenerator< T > & other) [inline]
```

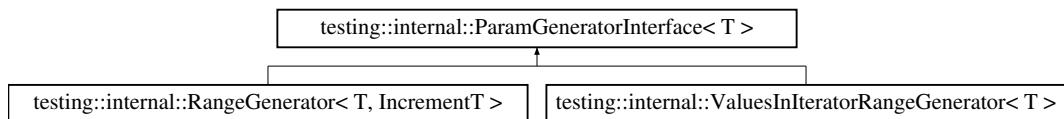
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.68 Dokumentacja szablonu klasy testing::internal::ParamGeneratorInterface< T >

```
#include <gtest-param-util.h>
```

Diagram dziedziczenia dla testing::internal::ParamGeneratorInterface< T >



Typy publiczne

- typedef T [ParamType](#)

Metody publiczne

- virtual `~ParamGeneratorInterface ()`
- virtual `ParamIteratorInterface< T > * Begin () const =0`
- virtual `ParamIteratorInterface< T > * End () const =0`

8.68.1 Dokumentacja składowych definicji typu

8.68.1.1 ParamType

```
template<typename T>
typedef T testing::internal::ParamGeneratorInterface< T >::ParamType
```

8.68.2 Dokumentacja konstruktora i destruktora

8.68.2.1 ~ParamGeneratorInterface()

```
template<typename T>
virtual testing::internal::ParamGeneratorInterface< T >::~ParamGeneratorInterface () [inline],  
[virtual]
```

8.68.3 Dokumentacja funkcji składowych

8.68.3.1 Begin()

```
template<typename T>
virtual ParamIteratorInterface< T > * testing::internal::ParamGeneratorInterface< T >::Begin  
() const [pure virtual]
```

Implementowany w `testing::internal::RangeGenerator< T, IncrementT >` i `testing::internal::ValuesIteratorRangeGenerator< T >`.

8.68.3.2 End()

```
template<typename T>
virtual ParamIteratorInterface< T > * testing::internal::ParamGeneratorInterface< T >::End ()  
const [pure virtual]
```

Implementowany w `testing::internal::RangeGenerator< T, IncrementT >` i `testing::internal::ValuesIteratorRangeGenerator< T >`.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.69 Dokumentacja szablonu klasy testing::internal::ParamIterator< T >

```
#include <gtest-param-util.h>
```

Typy publiczne

- `typedef T value_type`
- `typedef const T & reference`
- `typedef ptrdiff_t difference_type`

Metody publiczne

- `ParamIterator (const ParamIterator &other)`
- `ParamIterator & operator= (const ParamIterator &other)`
- `const T & operator* () const`
- `const T * operator> () const`
- `ParamIterator & operator++ ()`
- `ParamIterator operator++ (int)`
- `bool operator== (const ParamIterator &other) const`
- `bool operator!= (const ParamIterator &other) const`

Przyjaciele

- `class ParamGenerator< T >`

8.69.1 Dokumentacja składowych definicji typu

8.69.1.1 difference_type

```
template<typename T>
typedef ptrdiff_t testing::internal::ParamIterator< T >::difference_type
```

8.69.1.2 reference

```
template<typename T>
typedef const T& testing::internal::ParamIterator< T >::reference
```

8.69.1.3 value_type

```
template<typename T>
typedef T testing::internal::ParamIterator< T >::value_type
```

8.69.2 Dokumentacja konstruktora i destruktora

8.69.2.1 ParamIterator()

```
template<typename T>
testing::internal::ParamIterator< T >::ParamIterator (
    const ParamIterator< T > & other) [inline]
```

8.69.3 Dokumentacja funkcji składowych

8.69.3.1 operator"!=()

```
template<typename T>
bool testing::internal::ParamIterator< T >::operator!= (
    const ParamIterator< T > & other) const [inline]
```

8.69.3.2 operator*()

```
template<typename T>
const T & testing::internal::ParamIterator< T >::operator* () const [inline]
```

8.69.3.3 operator++() [1/2]

```
template<typename T>
ParamIterator & testing::internal::ParamIterator< T >::operator++ () [inline]
```

8.69.3.4 operator++() [2/2]

```
template<typename T>
ParamIterator testing::internal::ParamIterator< T >::operator++ (
    int ) [inline]
```

8.69.3.5 operator->()

```
template<typename T>
const T * testing::internal::ParamIterator< T >::operator-> () const [inline]
```

8.69.3.6 operator=()

```
template<typename T>
ParamIterator & testing::internal::ParamIterator< T >::operator= (
    const ParamIterator< T > & other) [inline]
```

8.69.3.7 operator==()

```
template<typename T>
bool testing::internal::ParamIterator< T >::operator== (
    const ParamIterator< T > & other) const [inline]
```

8.69.4 Dokumentacja przyjaciół i powiązanych symboli

8.69.4.1 ParamGenerator< T >

```
template<typename T>
friend class ParamGenerator< T > [friend]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.70 Dokumentacja szablonu klasy testing::internal::ParamIteratorInterface< T >

```
#include <gtest-param-util.h>
```

Metody publiczne

- virtual ~ParamIteratorInterface ()
- virtual const ParamIteratorInterface< T > * BaseGenerator () const =0
- virtual void Advance ()=0
- virtual ParamIteratorInterface * Clone () const =0
- virtual const T * Current () const =0
- virtual bool Equals (const ParamIteratorInterface &other) const =0

8.70.1 Dokumentacja konstruktora i destruktora

8.70.1.1 ~ParamIteratorInterface()

```
template<typename T>
virtual testing::internal::ParamIteratorInterface< T >::~ParamIteratorInterface () [inline],  
[virtual]
```

8.70.2 Dokumentacja funkcji składowych

8.70.2.1 Advance()

```
template<typename T>
virtual void testing::internal::ParamIteratorInterface< T >::Advance () [pure virtual]
```

8.70.2.2 BaseGenerator()

```
template<typename T>
virtual const ParamGeneratorInterface< T > * testing::internal::ParamIteratorInterface< T >::  
BaseGenerator () const [pure virtual]
```

8.70.2.3 Clone()

```
template<typename T>
virtual ParamIteratorInterface * testing::internal::ParamIteratorInterface< T >::Clone () const [pure virtual]
```

8.70.2.4 Current()

```
template<typename T>
virtual const T * testing::internal::ParamIteratorInterface< T >::Current () const [pure virtual]
```

8.70.2.5 Equals()

```
template<typename T>
virtual bool testing::internal::ParamIteratorInterface< T >::Equals (const ParamIteratorInterface< T > & other) const [pure virtual]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.71 Dokumentacja szablonu struktury testing::internal::ParamNameGenFunc< ParamType >

```
#include <gtest-param-util.h>
```

Typy publiczne

- `typedef std::string Type(const TestParamInfo< ParamType > &)`

8.71.1 Dokumentacja składowych definicji typu

8.71.1.1 Type

```
template<class ParamType>
typedef std::string testing::internal::ParamNameGenFunc< ParamType >::Type (const TestParamInfo< ParamType > &)
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.72 Dokumentacja struktury testing::PrintToStringParamName

```
#include <gtest-param-util.h>
```

Metody publiczne

- template<class ParamType>
std::string **operator()** (const **TestParamInfo**< ParamType > &info) const

8.72.1 Dokumentacja funkcji składowych

8.72.1.1 operator()()

```
template<class ParamType>
std::string testing::PrintToStringParamName::operator() (
    const TestParamInfo< ParamType > & info) const [inline]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.73 Dokumentacja klasy testing::internal::Random

```
#include <gtest-internal.h>
```

Metody publiczne

- **Random** (**UInt32** seed)
- void **Reseed** (**UInt32** seed)
- **UInt32 Generate** (**UInt32** range)

Statyczne atrybuty publiczne

- static const **UInt32 kMaxRange** = 1u << 31

8.73.1 Dokumentacja konstruktora i destruktora

8.73.1.1 Random()

```
testing::internal::Random::Random (
    UInt32 seed) [inline], [explicit]
```

8.73.2 Dokumentacja funkcji składowych

8.73.2.1 Generate()

```
UInt32 testing::internal::Random::Generate (
    UInt32 range)
```

8.73.2.2 Reseed()

```
void testing::internal::Random::Reseed (
    UInt32 seed) [inline]
```

8.73.3 Dokumentacja atrybutów składowych

8.73.3.1 kMaxRange

```
const UInt32 testing::internal::Random::kMaxRange = 1u << 31 [static]
```

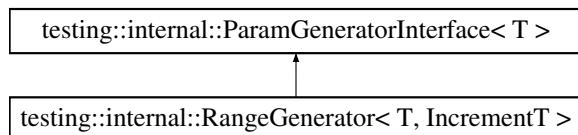
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.74 Dokumentacja szablonu klasy testing::internal::RangeGenerator< T, IncrementT >

```
#include <gtest-param-util.h>
```

Diagram dziedziczenia dla testing::internal::RangeGenerator< T, IncrementT >



Metody publiczne

- [RangeGenerator](#) (T begin, T end, IncrementT step)
- virtual [~RangeGenerator](#) ()
- virtual [ParamIteratorInterface< T > * Begin](#) () const
- virtual [ParamIteratorInterface< T > * End](#) () const

Metody publiczne dziedziczone z [testing::internal::ParamGeneratorInterface< T >](#)

- virtual [~ParamGeneratorInterface](#) ()

Dodatkowe dziedziczone składowe

Typy publiczne dziedziczone z testing::internal::ParamGeneratorInterface< T >

- typedef T ParamType

8.74.1 Dokumentacja konstruktora i destruktora

8.74.1.1 RangeGenerator()

```
template<typename T, typename IncrementT>
testing::internal::RangeGenerator< T, IncrementT >::RangeGenerator (
    T begin,
    T end,
    IncrementT step) [inline]
```

8.74.1.2 ~RangeGenerator()

```
template<typename T, typename IncrementT>
virtual testing::internal::RangeGenerator< T, IncrementT >::~RangeGenerator () [inline],
[virtual]
```

8.74.2 Dokumentacja funkcji składowych

8.74.2.1 Begin()

```
template<typename T, typename IncrementT>
virtual ParamIteratorInterface< T > * testing::internal::RangeGenerator< T, IncrementT >::←
Begin () const [inline], [virtual]
```

Implementuje testing::internal::ParamGeneratorInterface< T >.

8.74.2.2 End()

```
template<typename T, typename IncrementT>
virtual ParamIteratorInterface< T > * testing::internal::RangeGenerator< T, IncrementT >::End
() const [inline], [virtual]
```

Implementuje testing::internal::ParamGeneratorInterface< T >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.75 Dokumentacja klasy testing::internal::RE

```
#include <gtest-port.h>
```

Metody publiczne

- `RE (const RE &other)`
- `RE (const ::std::string ®ex)`
- `RE (const char *regex)`
- `~RE ()`
- `const char * pattern () const`

Statyczne metody publiczne

- `static bool FullMatch (const ::std::string &str, const RE &re)`
- `static bool PartialMatch (const ::std::string &str, const RE &re)`
- `static bool FullMatch (const char *str, const RE &re)`
- `static bool PartialMatch (const char *str, const RE &re)`

8.75.1 Dokumentacja konstruktora i destruktora

8.75.1.1 RE() [1/3]

```
testing::internal::RE::RE (
    const RE & other)  [inline]
```

8.75.1.2 RE() [2/3]

```
testing::internal::RE::RE (
    const ::std::string & regex)  [inline]
```

8.75.1.3 RE() [3/3]

```
testing::internal::RE::RE (
    const char * regex)  [inline]
```

8.75.1.4 ~RE()

```
testing::internal::RE::~RE ()
```

8.75.2 Dokumentacja funkcji składowych

8.75.2.1 FullMatch() [1/2]

```
bool testing::internal::RE::FullMatch (
    const ::std::string & str,
    const RE & re)  [inline], [static]
```

8.75.2.2 FullMatch() [2/2]

```
bool testing::internal::RE::FullMatch (
    const char * str,
    const RE & re) [static]
```

8.75.2.3 PartialMatch() [1/2]

```
bool testing::internal::RE::PartialMatch (
    const ::std::string & str,
    const RE & re) [inline], [static]
```

8.75.2.4 PartialMatch() [2/2]

```
bool testing::internal::RE::PartialMatch (
    const char * str,
    const RE & re) [static]
```

8.75.2.5 pattern()

```
const char * testing::internal::RE::pattern () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.76 Dokumentacja struktury testing::internal::RelationToSourceCopy

```
#include <gtest-internal.h>
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.77 Dokumentacja struktury testing::internal::RelationToSourceReference

```
#include <gtest-internal.h>
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.78 Dokumentacja szablonu struktury testing::internal::RemoveConst< T >

```
#include <gtest-internal.h>
```

Typy publiczne

- `typedef T type`

8.78.1 Dokumentacja składowych definicji typu

8.78.1.1 type

```
template<typename T>
typedef T testing::internal::RemoveConst< T >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.79 Dokumentacja szablonu struktury testing::internal::RemoveConst< const T >

```
#include <gtest-internal.h>
```

Typy publiczne

- `typedef T type`

8.79.1 Dokumentacja składowych definicji typu

8.79.1.1 type

```
template<typename T>
typedef T testing::internal::RemoveConst< const T >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.80 Dokumentacja szablonu struktury testing::internal::RemoveConst< const T[N]>

```
#include <gtest-internal.h>
```

Typy publiczne

- `typedef RemoveConst< T >::type type[N]`

8.80.1 Dokumentacja składowych definicji typu

8.80.1.1 type

```
template<typename T, size_t N>
typedef RemoveConst<T>::type testing::internal::RemoveConst< const T[N]>::type[N]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.81 Dokumentacja szablonu struktury testing::internal::RemoveReference< T >

```
#include <gtest-internal.h>
```

Typy publiczne

- `typedef T type`

8.81.1 Dokumentacja składowych definicji typu

8.81.1.1 type

```
template<typename T>
typedef T testing::internal::RemoveReference< T >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.82 Dokumentacja szablonu struktury `testing::internal::RemoveReference< T & >`

```
#include <gtest-internal.h>
```

Typy publiczne

- `typedef T type`

8.82.1 Dokumentacja składowych definicji typu

8.82.1.1 `type`

```
template<typename T>
typedef T testing::internal::RemoveReference< T & >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.83 Dokumentacja szablonu struktury `testing::internal::RvalueRef< T >`

```
#include <gtest-port.h>
```

Typy publiczne

- `typedef const T & type`

8.83.1 Dokumentacja składowych definicji typu

8.83.1.1 `type`

```
template<typename T>
typedef const T& testing::internal::RvalueRef< T >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.84 Dokumentacja szablonu struktury

std::tr1::gtest_internal::SameSizeTuplePrefixComparator< kSize1, kSize2 >

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.85 Dokumentacja struktury

std::tr1::gtest_internal::SameSizeTuplePrefixComparator< 0, 0 >

```
#include <gtest-tuple.h>
```

Statyczne metody publiczne

- template<class Tuple1, class Tuple2>
static bool **Eq** (const Tuple1 &, const Tuple2 &)

8.85.1 Dokumentacja funkcji składowych

8.85.1.1 Eq()

```
template<class Tuple1, class Tuple2>
bool std::tr1::gtest_internal::SameSizeTuplePrefixComparator< 0, 0 >::Eq (
    const Tuple1 & ,
    const Tuple2 & ) [inline], [static]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.86 Dokumentacja szablonu struktury

std::tr1::gtest_internal::SameSizeTuplePrefixComparator< k, k >

```
#include <gtest-tuple.h>
```

Statyczne metody publiczne

- template<class Tuple1, class Tuple2>
static bool **Eq** (const Tuple1 &t1, const Tuple2 &t2)

8.86.1 Dokumentacja funkcji składowych

8.86.1.1 Eq()

```
template<int k>
template<class Tuple1, class Tuple2>
bool std::tr1::gtest_internal::SameSizeTuplePrefixComparator<k, k>::Eq (
    const Tuple1 & t1,
    const Tuple2 & t2) [inline], [static]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.87 Dokumentacja szablonu klasy testing::internal::scoped_ptr< T >

```
#include <gtest-port.h>
```

Typy publiczne

- typedef T `element_type`

Metody publiczne

- `scoped_ptr` (T *p=NULL)
- `~scoped_ptr` ()
- T & `operator*` () const
- T * `operator->` () const
- T * `get` () const
- T * `release` ()
- void `reset` (T *p=NULL)

Przyjaciele

- void `swap` (scoped_ptr &a, scoped_ptr &b)

8.87.1 Dokumentacja składowych definicji typu

8.87.1.1 element_type

```
template<typename T>
typedef T testing::internal::scoped_ptr< T >::element_type
```

8.87.2 Dokumentacja konstruktora i destruktora

8.87.2.1 scoped_ptr()

```
template<typename T>
testing::internal::scoped_ptr< T >::scoped_ptr (
    T * p = NULL) [inline], [explicit]
```

8.87.2.2 ~scoped_ptr()

```
template<typename T>
testing::internal::scoped_ptr< T >::~scoped_ptr () [inline]
```

8.87.3 Dokumentacja funkcji składowych

8.87.3.1 get()

```
template<typename T>
T * testing::internal::scoped_ptr< T >::get () const [inline]
```

8.87.3.2 operator*()

```
template<typename T>
T & testing::internal::scoped_ptr< T >::operator* () const [inline]
```

8.87.3.3 operator->()

```
template<typename T>
T * testing::internal::scoped_ptr< T >::operator-> () const [inline]
```

8.87.3.4 release()

```
template<typename T>
T * testing::internal::scoped_ptr< T >::release () [inline]
```

8.87.3.5 reset()

```
template<typename T>
void testing::internal::scoped_ptr< T >::reset (
    T * p = NULL) [inline]
```

8.87.4 Dokumentacja przyjaciół i powiązanych symboli

8.87.4.1 swap

```
template<typename T>
void swap (
    scoped_ptr< T > & a,
    scoped_ptr< T > & b) [friend]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.88 Dokumentacja klasy testing::ScopedTrace

```
#include <gtest.h>
```

Metody publiczne

- template<typename T>
`ScopedTrace` (const char *file, int line, const T &message)
- `ScopedTrace` (const char *file, int line, const char *message)
- `ScopedTrace` (const char *file, int line, const std::string &message)
- `~ScopedTrace` ()

8.88.1 Dokumentacja konstruktora i destruktora

8.88.1.1 ScopedTrace() [1/3]

```
template<typename T>
testing::ScopedTrace::ScopedTrace (
    const char * file,
    int line,
    const T & message) [inline]
```

8.88.1.2 ScopedTrace() [2/3]

```
testing::ScopedTrace::ScopedTrace (
    const char * file,
    int line,
    const char * message) [inline]
```

8.88.1.3 ScopedTrace() [3/3]

```
testing::ScopedTrace::ScopedTrace (
    const char * file,
    int line,
    const std::string & message) [inline]
```

8.88.1.4 ~ScopedTrace()

```
testing::ScopedTrace::~ScopedTrace ()
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

8.89 Dokumentacja szablonu struktury testing::internal::StaticAssertTypeEqHelper< T1, T2 >

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.90 Dokumentacja szablonu struktury testing::internal::StaticAssertTypeEqHelper< T, T >

```
#include <gtest-port.h>
```

Typy publiczne

- enum { **value** = true }

8.90.1 Dokumentacja składowych wyliczanych

8.90.1.1 anonymous enum

```
template<typename T>
anonymous enum
```

Wartości wyliczeń

value	
-------	--

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.91 Dokumentacja klasy testing::internal::String

```
#include <gtest-string.h>
```

Statyczne metody publiczne

- static const char * [CloneCString](#) (const char *c_str)
- static bool [CStringEquals](#) (const char *lhs, const char *rhs)
- static std::string [ShowWideCString](#) (const wchar_t *wide_c_str)
- static bool [WideCStringEquals](#) (const wchar_t *lhs, const wchar_t *rhs)
- static bool [CaseInsensitiveCStringEquals](#) (const char *lhs, const char *rhs)
- static bool [CaseInsensitiveWideCStringEquals](#) (const wchar_t *lhs, const wchar_t *rhs)
- static bool [EndsWithCaseInsensitive](#) (const std::string &str, const std::string &suffix)
- static std::string [FormatIntWidth2](#) (int value)
- static std::string [FormatHexInt](#) (int value)
- static std::string [FormatByte](#) (unsigned char value)

8.91.1 Dokumentacja funkcji składowych

8.91.1.1 CaseInsensitiveCStringEquals()

```
bool testing::internal::String::CaseInsensitiveCStringEquals (
    const char * lhs,
    const char * rhs)  [static]
```

8.91.1.2 CaseInsensitiveWideCStringEquals()

```
bool testing::internal::String::CaseInsensitiveWideCStringEquals (
    const wchar_t * lhs,
    const wchar_t * rhs)  [static]
```

8.91.1.3 CloneCString()

```
const char * testing::internal::String::CloneCString (
    const char * c_str)  [static]
```

8.91.1.4 CStringEquals()

```
bool testing::internal::String::CStringEquals (
    const char * lhs,
    const char * rhs)  [static]
```

8.91.1.5 EndsWithCaseInsensitive()

```
bool testing::internal::String::EndsWithCaseInsensitive (
    const std::string & str,
    const std::string & suffix)  [static]
```

8.91.1.6 FormatByte()

```
std::string testing::internal::String::FormatByte (
    unsigned char value) [static]
```

8.91.1.7 FormatHexInt()

```
std::string testing::internal::String::FormatHexInt (
    int value) [static]
```

8.91.1.8 FormatIntWidth2()

```
std::string testing::internal::String::FormatIntWidth2 (
    int value) [static]
```

8.91.1.9 ShowWideCString()

```
std::string testing::internal::String::ShowWideCString (
    const wchar_t * wide_c_str) [static]
```

8.91.1.10 WideCStringEquals()

```
bool testing::internal::String::WideCStringEquals (
    const wchar_t * lhs,
    const wchar_t * rhs) [static]
```

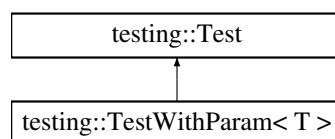
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-string.h

8.92 Dokumentacja klasy testing::Test

```
#include <gtest.h>
```

Diagram dziedziczenia dla testing::Test



Typy publiczne

- typedef internal::SetUpTestCaseFunc SetUpTestCaseFunc
- typedef internal::TearDownTestCaseFunc TearDownTestCaseFunc

Metody publiczne

- virtual `~Test ()`

Statyczne metody publiczne

- static void `SetUpTestCase ()`
- static void `TearDownTestCase ()`
- static bool `HasFatalFailure ()`
- static bool `HasNonfatalFailure ()`
- static bool `HasFailure ()`
- static void `RecordProperty (const std::string &key, const std::string &value)`
- static void `RecordProperty (const std::string &key, int value)`

Metody chronione

- `Test ()`
- virtual void `SetUp ()`
- virtual void `TearDown ()`

Przyjaciele

- class `TestInfo`

8.92.1 Dokumentacja składowych definicji typu

8.92.1.1 SetUpTestCaseFunc

```
typedef internal::SetUpTestCaseFunc testing::Test::SetUpTestCaseFunc
```

8.92.1.2 TearDownTestCaseFunc

```
typedef internal::TearDownTestCaseFunc testing::Test::TearDownTestCaseFunc
```

8.92.2 Dokumentacja konstruktora i destruktora

8.92.2.1 ~Test()

```
virtual testing::Test::~Test () [virtual]
```

8.92.2.2 Test()

```
testing::Test::Test () [protected]
```

8.92.3 Dokumentacja funkcji składowych

8.92.3.1 HasFailure()

```
bool testing::Test::HasFailure () [inline], [static]
```

8.92.3.2 HasFatalFailure()

```
bool testing::Test::HasFatalFailure () [static]
```

8.92.3.3 HasNonfatalFailure()

```
bool testing::Test::HasNonfatalFailure () [static]
```

8.92.3.4 RecordProperty() [1/2]

```
void testing::Test::RecordProperty (
    const std::string & key,
    const std::string & value) [static]
```

8.92.3.5 RecordProperty() [2/2]

```
void testing::Test::RecordProperty (
    const std::string & key,
    int value) [static]
```

8.92.3.6 SetUp()

```
virtual void testing::Test::SetUp () [protected], [virtual]
```

8.92.3.7 SetUpTestCase()

```
void testing::Test::SetUpTestCase () [inline], [static]
```

8.92.3.8 TearDown()

```
virtual void testing::Test::TearDown () [protected], [virtual]
```

8.92.3.9 TearDownTestCase()

```
void testing::Test::TearDownTestCase () [inline], [static]
```

8.92.4 Dokumentacja przyjaciół i powiązanych symboli

8.92.4.1 TestInfo

```
friend class TestInfo [friend]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

8.93 Dokumentacja klasy testing::TestCase

```
#include <gtest.h>
```

Metody publiczne

- `TestCase` (const char *`name`, const char *`a_type_param`, `Test::SetUpTestCaseFunc` `set_up_tc`, `Test::TearDownTestCaseFunc` `tear_down_tc`)
- virtual ~`TestCase` ()
- const char * `name` () const
- const char * `type_param` () const
- bool `should_run` () const
- int `successful_test_count` () const
- int `failed_test_count` () const
- int `reportable_disabled_test_count` () const
- int `disabled_test_count` () const
- int `reportable_test_count` () const
- int `test_to_run_count` () const
- int `total_test_count` () const
- bool `Passed` () const
- bool `Failed` () const
- `TimeInMillis elapsed_time` () const
- const `TestInfo *` `GetTestInfo` (int i) const
- const `TestResult` & `ad_hoc_test_result` () const

Przyjaciele

- class `Test`
- class `internal::UnitTestImpl`

8.93.1 Dokumentacja konstruktora i destruktora

8.93.1.1 TestCase()

```
testing::TestCase::TestCase (
    const char * name,
    const char * a_type_param,
    Test::SetUpTestCaseFunc set_up_tc,
    Test::TearDownTestCaseFunc tear_down_tc)
```

8.93.1.2 ~TestCase()

```
virtual testing::TestCase::~TestCase () [virtual]
```

8.93.2 Dokumentacja funkcji składowych

8.93.2.1 ad_hoc_test_result()

```
const TestResult & testing::TestCase::ad_hoc_test_result () const [inline]
```

8.93.2.2 disabled_test_count()

```
int testing::TestCase::disabled_test_count () const
```

8.93.2.3 elapsed_time()

```
TimeInMillis testing::TestCase::elapsed_time () const [inline]
```

8.93.2.4 Failed()

```
bool testing::TestCase::Failed () const [inline]
```

8.93.2.5 failed_test_count()

```
int testing::TestCase::failed_test_count () const
```

8.93.2.6 GetTestInfo()

```
const TestInfo * testing::TestCase::GetTestInfo (
    int i) const
```

8.93.2.7 name()

```
const char * testing::TestCase::name () const [inline]
```

8.93.2.8 Passed()

```
bool testing::TestCase::Passed () const [inline]
```

8.93.2.9 reportable_disabled_test_count()

```
int testing::TestCase::reportable_disabled_test_count () const
```

8.93.2.10 reportable_test_count()

```
int testing::TestCase::reportable_test_count () const
```

8.93.2.11 should_run()

```
bool testing::TestCase::should_run () const [inline]
```

8.93.2.12 successful_test_count()

```
int testing::TestCase::successful_test_count () const
```

8.93.2.13 test_to_run_count()

```
int testing::TestCase::test_to_run_count () const
```

8.93.2.14 total_test_count()

```
int testing::TestCase::total_test_count () const
```

8.93.2.15 type_param()

```
const char * testing::TestCase::type_param () const [inline]
```

8.93.3 Dokumentacja przyjaciół i powiązanych symboli

8.93.3.1 internal::UnitTestImpl

```
friend class internal::UnitTestImpl [friend]
```

8.93.3.2 Test

```
friend class Test [friend]
```

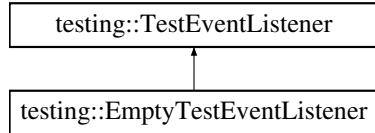
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

8.94 Dokumentacja klasy testing::TestEventListener

```
#include <gtest.h>
```

Diagram dziedziczenia dla testing::TestEventListener



Metody publiczne

- virtual [~TestEventListener \(\)](#)
- virtual void [OnTestProgramStart \(const UnitTest &unit_test\)=0](#)
- virtual void [OnTestIterationStart \(const UnitTest &unit_test, int iteration\)=0](#)
- virtual void [OnEnvironmentsSetUpStart \(const UnitTest &unit_test\)=0](#)
- virtual void [OnEnvironmentsSetUpEnd \(const UnitTest &unit_test\)=0](#)
- virtual void [OnTestCaseStart \(const TestCase &test_case\)=0](#)
- virtual void [OnTestStart \(const TestInfo &test_info\)=0](#)
- virtual void [OnTestPartResult \(const TestPartResult &test_part_result\)=0](#)
- virtual void [OnTestEnd \(const TestInfo &test_info\)=0](#)
- virtual void [OnTestCaseEnd \(const TestCase &test_case\)=0](#)
- virtual void [OnEnvironmentsTearDownStart \(const UnitTest &unit_test\)=0](#)
- virtual void [OnEnvironmentsTearDownEnd \(const UnitTest &unit_test\)=0](#)
- virtual void [OnTestIterationEnd \(const UnitTest &unit_test, int iteration\)=0](#)
- virtual void [OnTestProgramEnd \(const UnitTest &unit_test\)=0](#)

8.94.1 Dokumentacja konstruktora i destruktora

8.94.1.1 ~TestEventListener()

```
virtual testing::TestEventListener::~TestEventListener () [inline], [virtual]
```

8.94.2 Dokumentacja funkcji składowych

8.94.2.1 OnEnvironmentsSetUpEnd()

```
virtual void testing::TestEventListener::OnEnvironmentsSetUpEnd (
    const UnitTest & unit_test) [pure virtual]
```

Implementowany w [testing::EmptyTestEventListener](#).

8.94.2.2 OnEnvironmentsSetUpStart()

```
virtual void testing::TestEventListener::OnEnvironmentsSetUpStart (
    const UnitTest & unit_test) [pure virtual]
```

Implementowany w [testing::EmptyTestEventListener](#).

8.94.2.3 OnEnvironmentsTearDownEnd()

```
virtual void testing::TestEventListener::OnEnvironmentsTearDownEnd (
    const UnitTest & unit_test) [pure virtual]
```

Implementowany w [testing::EmptyTestEventListener](#).

8.94.2.4 OnEnvironmentsTearDownStart()

```
virtual void testing::TestEventListener::OnEnvironmentsTearDownStart (
    const UnitTest & unit_test) [pure virtual]
```

Implementowany w [testing::EmptyTestEventListener](#).

8.94.2.5 OnTestCaseEnd()

```
virtual void testing::TestEventListener::OnTestCaseEnd (
    const TestCase & test_case) [pure virtual]
```

Implementowany w [testing::EmptyTestEventListener](#).

8.94.2.6 OnTestCaseStart()

```
virtual void testing::TestEventListener::OnTestCaseStart (
    const TestCase & test_case) [pure virtual]
```

Implementowany w [testing::EmptyTestEventListener](#).

8.94.2.7 OnTestEnd()

```
virtual void testing::TestEventListener::OnTestEnd (
    const TestInfo & test_info) [pure virtual]
```

Implementowany w [testing::EmptyTestEventListener](#).

8.94.2.8 OnTestIterationEnd()

```
virtual void testing::TestEventListener::OnTestIterationEnd (
    const UnitTest & unit_test,
    int iteration) [pure virtual]
```

Implementowany w [testing::EmptyTestEventListener](#).

8.94.2.9 OnTestIterationStart()

```
virtual void testing::TestEventListener::OnTestIterationStart (
    const UnitTest & unit_test,
    int iteration) [pure virtual]
```

Implementowany w [testing::EmptyTestEventListener](#).

8.94.2.10 OnTestPartResult()

```
virtual void testing::TestEventListener::OnTestPartResult (
    const TestPartResult & test_part_result) [pure virtual]
```

Implementowany w [testing::EmptyTestEventListener](#).

8.94.2.11 OnTestProgramEnd()

```
virtual void testing::TestEventListener::OnTestProgramEnd (
    const UnitTest & unit_test) [pure virtual]
```

Implementowany w [testing::EmptyTestEventListener](#).

8.94.2.12 OnTestProgramStart()

```
virtual void testing::TestEventListener::OnTestProgramStart (
    const UnitTest & unit_test) [pure virtual]
```

Implementowany w [testing::EmptyTestEventListener](#).

8.94.2.13 OnTestStart()

```
virtual void testing::TestEventListener::OnTestStart (
    const TestInfo & test_info) [pure virtual]
```

Implementowany w [testing::EmptyTestEventListener](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

8.95 Dokumentacja klasy testing::TestEventListeners

```
#include <gtest.h>
```

Metody publiczne

- [TestEventListeners \(\)](#)
- [~TestEventListeners \(\)](#)
- [void Append \(TestEventListener *listener\)](#)
- [TestEventListener * Release \(TestEventListener *listener\)](#)
- [TestEventListener * default_result_printer \(\) const](#)
- [TestEventListener * default_xml_generator \(\) const](#)

Przyjaciele

- class [TestCase](#)
- class [TestInfo](#)
- class [internal::DefaultGlobalTestPartResultReporter](#)
- class [internal::NoExecDeathTest](#)
- class [internal::TestEventListenersAccessor](#)
- class [internal::UnitTestImpl](#)

8.95.1 Dokumentacja konstruktora i destruktora

8.95.1.1 [TestEventListeners\(\)](#)

```
testing:::TestEventListeners:::TestEventListeners ()
```

8.95.1.2 [~TestEventListeners\(\)](#)

```
testing:::TestEventListeners:::~TestEventListeners ()
```

8.95.2 Dokumentacja funkcji składowych

8.95.2.1 [Append\(\)](#)

```
void testing:::TestEventListeners:::Append (
    TestEventListener * listener)
```

8.95.2.2 [default_result_printer\(\)](#)

```
TestEventListener * testing:::TestEventListeners:::default_result_printer () const [inline]
```

8.95.2.3 [default_xml_generator\(\)](#)

```
TestEventListener * testing:::TestEventListeners:::default_xml_generator () const [inline]
```

8.95.2.4 [Release\(\)](#)

```
TestEventListener * testing:::TestEventListeners:::Release (
    TestEventListener * listener)
```

8.95.3 Dokumentacja przyjaciół i powiązanych symboli

8.95.3.1 [internal::DefaultGlobalTestPartResultReporter](#)

```
friend class internal::DefaultGlobalTestPartResultReporter [friend]
```

8.95.3.2 internal::NoExecDeathTest

```
friend class internal::NoExecDeathTest [friend]
```

8.95.3.3 internal::TestEventListenersAccessor

```
friend class internal::TestEventListenersAccessor [friend]
```

8.95.3.4 internal::UnitTestImpl

```
friend class internal::UnitTestImpl [friend]
```

8.95.3.5 TestCase

```
friend class TestCase [friend]
```

8.95.3.6 TestInfo

```
friend class TestInfo [friend]
```

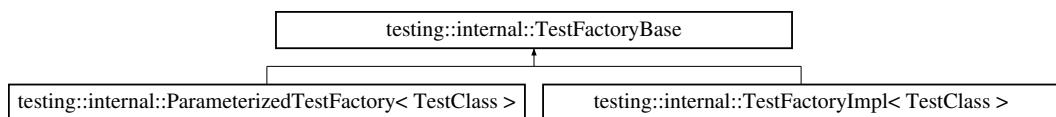
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

8.96 Dokumentacja klasy testing::internal::TestFactoryBase

```
#include <gtest-internal.h>
```

Diagram dziedziczenia dla testing::internal::TestFactoryBase



Metody publiczne

- virtual [~TestFactoryBase \(\)](#)
- virtual [Test * CreateTest \(\)=0](#)

Metody chronione

- [TestFactoryBase \(\)](#)

8.96.1 Dokumentacja konstruktora i destruktora

8.96.1.1 ~TestFactoryBase()

```
virtual testing::internal::TestFactoryBase::~TestFactoryBase () [inline], [virtual]
```

8.96.1.2 TestFactoryBase()

```
testing::internal::TestFactoryBase::TestFactoryBase () [inline], [protected]
```

8.96.2 Dokumentacja funkcji składowych

8.96.2.1 CreateTest()

```
virtual Test * testing::internal::TestFactoryBase::CreateTest () [pure virtual]
```

Implementowany w [testing::internal::ParameterizedTestFactory<TestClass>](#) i [testing::internal::TestFactoryImpl<TestClass>](#).

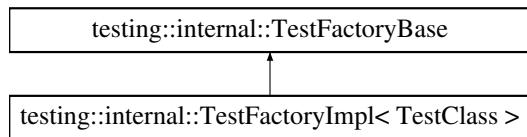
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.97 Dokumentacja szablonu klasy testing::internal::TestFactoryImpl<TestClass>

```
#include <gtest-internal.h>
```

Diagram dziedziczenia dla [testing::internal::TestFactoryImpl<TestClass>](#)



Metody publiczne

- virtual [Test * CreateTest\(\)](#)

Metody publiczne dziedziczone z [testing::internal::TestFactoryBase](#)

- virtual [~TestFactoryBase\(\)](#)

Dodatkowe dziedziczone składowe

Metody chronione dziedziczone z [testing::internal::TestFactoryBase](#)

- [TestFactoryBase \(\)](#)

8.97.1 Dokumentacja funkcji składowych

8.97.1.1 CreateTest()

```
template<class TestClass>
virtual Test * testing::internal::TestFactoryImpl<TestClass>::CreateTest () [inline], [virtual]
```

Implementuje [testing::internal::TestFactoryBase](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.98 Dokumentacja klasy testing::TestInfo

```
#include <gtest.h>
```

Metody publiczne

- [~TestInfo \(\)](#)
- const char * [test_case_name \(\) const](#)
- const char * [name \(\) const](#)
- const char * [type_param \(\) const](#)
- const char * [value_param \(\) const](#)
- const char * [file \(\) const](#)
- int [line \(\) const](#)
- bool [is_in_another_shard \(\) const](#)
- bool [should_run \(\) const](#)
- bool [is_reportable \(\) const](#)
- const TestResult * [result \(\) const](#)

Przyjaciele

- class [Test](#)
- class [TestCase](#)
- class [internal::UnitTestImpl](#)
- class [internal::StreamingListenerTest](#)
- TestInfo * [internal::MakeAndRegisterTestInfo \(const char *test_case_name, const char *name, const char *type_param, const char *value_param, internal::CodeLocation code_location, internal::Typeid fixture_class_id, Test::SetUpTestCaseFunc set_up_tc, Test::TearDownTestCaseFunc tear_down_tc, internal::TestFactoryBase *factory\)](#)

8.98.1 Dokumentacja konstruktora i destruktora

8.98.1.1 ~TestInfo()

```
testing::TestInfo::~TestInfo ()
```

8.98.2 Dokumentacja funkcji składowych

8.98.2.1 file()

```
const char * testing::TestInfo::file () const [inline]
```

8.98.2.2 is_in_another_shard()

```
bool testing::TestInfo::is_in_another_shard () const [inline]
```

8.98.2.3 is_reportable()

```
bool testing::TestInfo::is_reportable () const [inline]
```

8.98.2.4 line()

```
int testing::TestInfo::line () const [inline]
```

8.98.2.5 name()

```
const char * testing::TestInfo::name () const [inline]
```

8.98.2.6 result()

```
const TestResult * testing::TestInfo::result () const [inline]
```

8.98.2.7 should_run()

```
bool testing::TestInfo::should_run () const [inline]
```

8.98.2.8 test_case_name()

```
const char * testing::TestInfo::test_case_name () const [inline]
```

8.98.2.9 type_param()

```
const char * testing::TestInfo::type_param () const [inline]
```

8.98.2.10 value_param()

```
const char * testing::TestInfo::value_param () const [inline]
```

8.98.3 Dokumentacja przyjaciół i powiązanych symboli

8.98.3.1 internal::MakeAndRegisterTestInfo

```
TestInfo * internal::MakeAndRegisterTestInfo (
    const char * test_case_name,
    const char * name,
    const char * type_param,
    const char * value_param,
    internal::CodeLocation code_location,
    internal::TypeId fixture_class_id,
    Test::SetUpTestCaseFunc set_up_tc,
    Test::TearDownTestCaseFunc tear_down_tc,
    internal::TestFactoryBase * factory) [friend]
```

8.98.3.2 internal::StreamingListenerTest

```
friend class internal::StreamingListenerTest [friend]
```

8.98.3.3 internal::UnitTestImpl

```
friend class internal::UnitTestImpl [friend]
```

8.98.3.4 Test

```
friend class Test [friend]
```

8.98.3.5 TestCase

```
friend class TestCase [friend]
```

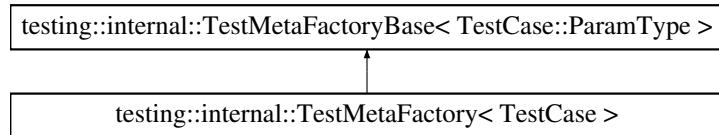
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

8.99 Dokumentacja szablonu klasy testing::internal::TestMetaFactory< TestCase >

```
#include <gtest-param-util.h>
```

Diagram dziedziczenia dla testing::internal::TestMetaFactory< TestCase >



Typy publiczne

- `typedef TestCase::ParamType ParamType`

Metody publiczne

- `TestMetaFactory ()`
- `virtual TestFactoryBase * CreateTestFactory (ParamType parameter)`

Metody publiczne dziedziczone z testing::internal::TestMetaFactoryBase< TestCase::ParamType >

- `virtual ~TestMetaFactory ()`
- `virtual TestFactoryBase * CreateTestFactory (ParamType parameter)=0`

8.99.1 Dokumentacja składowych definicji typu

8.99.1.1 ParamType

```
template<class TestCase>
typedef TestCase::ParamType testing::internal::TestMetaFactory< TestCase >::ParamType
```

8.99.2 Dokumentacja konstruktora i destruktora

8.99.2.1 TestMetaFactory()

```
template<class TestCase>
testing::internal::TestMetaFactory< TestCase >::TestMetaFactory () [inline]
```

8.99.3 Dokumentacja funkcji składowych

8.99.3.1 CreateTestFactory()

```
template<class TestCase>
virtual TestFactoryBase * testing::internal::TestMetaFactory< TestCase >::CreateTestFactory (
    ParamType parameter) [inline], [virtual]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.100 Dokumentacja szablonu klasy testing::internal::TestMetaFactoryBase< ParamType >

```
#include <gtest-param-util.h>
```

Metody publiczne

- virtual ~TestMetaFactoryBase ()
- virtual TestFactoryBase * CreateTestFactory (ParamType parameter)=0

8.100.1 Dokumentacja konstruktora i destruktora

8.100.1.1 ~TestMetaFactoryBase()

```
template<class ParamType>
virtual testing::internal::TestMetaFactoryBase< ParamType >::~TestMetaFactoryBase () [inline],
[virtual]
```

8.100.2 Dokumentacja funkcji składowych

8.100.2.1 CreateTestFactory()

```
template<class ParamType>
virtual TestFactoryBase * testing::internal::TestMetaFactoryBase< ParamType >::CreateTestFactory (
    ParamType parameter) [pure virtual]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.101 Dokumentacja szablonu struktury testing::TestParamInfo<ParamType >

```
#include <gtest-param-util.h>
```

Metody publiczne

- [TestParamInfo](#) (const ParamType &a_param, size_t an_index)

Atrybuty publiczne

- ParamType [param](#)
- size_t [index](#)

8.101.1 Dokumentacja konstruktora i destruktora

8.101.1.1 TestParamInfo()

```
template<class ParamType>
testing::TestParamInfo< ParamType >::TestParamInfo (
    const ParamType & a_param,
    size_t an_index) [inline]
```

8.101.2 Dokumentacja atrybutów składowych

8.101.2.1 index

```
template<class ParamType>
size_t testing::TestParamInfo< ParamType >::index
```

8.101.2.2 param

```
template<class ParamType>
ParamType testing::TestParamInfo< ParamType >::param
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-param-util.h](#)

8.102 Dokumentacja klasy testing::TestProperty

```
#include <gtest.h>
```

Metody publiczne

- `TestProperty` (const std::string &a_key, const std::string &a_value)
- const char * `key` () const
- const char * `value` () const
- void `SetValue` (const std::string &new_value)

8.102.1 Dokumentacja konstruktora i destruktora

8.102.1.1 `TestProperty()`

```
testing::TestProperty::TestProperty (
    const std::string & a_key,
    const std::string & a_value) [inline]
```

8.102.2 Dokumentacja funkcji składowych

8.102.2.1 `key()`

```
const char * testing::TestProperty::key () const [inline]
```

8.102.2.2 `SetValue()`

```
void testing::TestProperty::SetValue (
    const std::string & new_value) [inline]
```

8.102.2.3 `value()`

```
const char * testing::TestProperty::value () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

8.103 Dokumentacja klasy testing::TestResult

```
#include <gtest.h>
```

Metody publiczne

- `TestResult ()`
- `~TestResult ()`
- `int total_part_count () const`
- `int test_property_count () const`
- `bool Passed () const`
- `bool Failed () const`
- `bool HasFatalFailure () const`
- `bool HasNonfatalFailure () const`
- `TimeInMillis elapsed_time () const`
- `const TestPartResult & GetTestPartResult (int i) const`
- `const TestProperty & GetTestProperty (int i) const`

Przyjaciele

- class `TestInfo`
- class `TestCase`
- class `UnitTest`
- class `internal::DefaultGlobalTestPartResultReporter`
- class `internal::ExecDeathTest`
- class `internal::TestResultAccessor`
- class `internal::UnitTestImpl`
- class `internal::WindowsDeathTest`
- class `internal::FuchsiaDeathTest`

8.103.1 Dokumentacja konstruktora i destruktora

8.103.1.1 `TestResult()`

```
testing::TestResult::TestResult ()
```

8.103.1.2 `~TestResult()`

```
testing::TestResult::~TestResult ()
```

8.103.2 Dokumentacja funkcji składowych

8.103.2.1 `elapsed_time()`

```
TimeInMillis testing::TestResult::elapsed_time () const [inline]
```

8.103.2.2 `Failed()`

```
bool testing::TestResult::Failed () const
```

8.103.2.3 GetTestPartResult()

```
const TestPartResult & testing::TestResult::GetTestPartResult (
    int i) const
```

8.103.2.4 GetTestProperty()

```
const TestProperty & testing::TestResult::GetTestProperty (
    int i) const
```

8.103.2.5 HasFatalFailure()

```
bool testing::TestResult::HasFatalFailure () const
```

8.103.2.6 HasNonfatalFailure()

```
bool testing::TestResult::HasNonfatalFailure () const
```

8.103.2.7 Passed()

```
bool testing::TestResult::Passed () const [inline]
```

8.103.2.8 test_property_count()

```
int testing::TestResult::test_property_count () const
```

8.103.2.9 total_part_count()

```
int testing::TestResult::total_part_count () const
```

8.103.3 Dokumentacja przyjaciół i powiązanych symboli**8.103.3.1 internal::DefaultGlobalTestPartResultReporter**

```
friend class internal::DefaultGlobalTestPartResultReporter [friend]
```

8.103.3.2 internal::ExecDeathTest

```
friend class internal::ExecDeathTest [friend]
```

8.103.3.3 internal::FuchsiaDeathTest

```
friend class internal::FuchsiaDeathTest [friend]
```

8.103.3.4 internal::TestResultAccessor

```
friend class internal::TestResultAccessor [friend]
```

8.103.3.5 internal::UnitTestImpl

```
friend class internal::UnitTestImpl [friend]
```

8.103.3.6 internal::WindowsDeathTest

```
friend class internal::WindowsDeathTest [friend]
```

8.103.3.7 TestCase

```
friend class TestCase [friend]
```

8.103.3.8 TestInfo

```
friend class TestInfo [friend]
```

8.103.3.9 UnitTest

```
friend class UnitTest [friend]
```

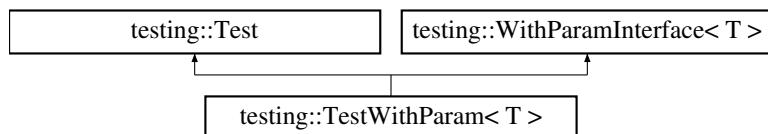
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

8.104 Dokumentacja szablonu klasy testing::TestWithParam< T >

```
#include <gtest.h>
```

Diagram dziedziczenia dla testing::TestWithParam< T >



Dodatkowe dziedziczone składowe

Typy publiczne dziedziczone z `testing::Test`

- `typedef internal::SetUpTestCaseFunc SetUpTestCaseFunc`
- `typedef internal::TearDownTestCaseFunc TearDownTestCaseFunc`

Typy publiczne dziedziczone z `testing::WithParamInterface< T >`

- `typedef T ParamType`

Metody publiczne dziedziczone z `testing::Test`

- `virtual ~Test ()`

Metody publiczne dziedziczone z `testing::WithParamInterface< T >`

- `virtual ~WithParamInterface ()`
- `const ParamType & GetParam () const`

Statyczne metody publiczne dziedziczone z `testing::Test`

- `static void SetUpTestCase ()`
- `static void TearDownTestCase ()`
- `static bool HasFatalFailure ()`
- `static bool HasNonfatalFailure ()`
- `static bool HasFailure ()`
- `static void RecordProperty (const std::string &key, const std::string &value)`
- `static void RecordProperty (const std::string &key, int value)`

Metody chronione dziedziczone z `testing::Test`

- `Test ()`
- `virtual void SetUp ()`
- `virtual void TearDown ()`

Dokumentacja dla tej klasy została wygenerowana z pliku:

- `packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h`

8.105 Dokumentacja szablonu klasy testing::internal::ThreadLocal< T >

```
#include <gtest-port.h>
```

Metody publiczne

- `ThreadLocal ()`
- `ThreadLocal (const T &value)`
- `T * pointer ()`
- `const T * pointer () const`
- `const T & get () const`
- `void set (const T &value)`

8.105.1 Dokumentacja konstruktora i destruktora**8.105.1.1 ThreadLocal() [1/2]**

```
template<typename T>
testing::internal::ThreadLocal< T >::ThreadLocal () [inline]
```

8.105.1.2 ThreadLocal() [2/2]

```
template<typename T>
testing::internal::ThreadLocal< T >::ThreadLocal (
    const T & value) [inline], [explicit]
```

8.105.2 Dokumentacja funkcji składowych**8.105.2.1 get()**

```
template<typename T>
const T & testing::internal::ThreadLocal< T >::get () const [inline]
```

8.105.2.2 pointer() [1/2]

```
template<typename T>
T * testing::internal::ThreadLocal< T >::pointer () [inline]
```

8.105.2.3 pointer() [2/2]

```
template<typename T>
const T * testing::internal::ThreadLocal< T >::pointer () const [inline]
```

8.105.2.4 set()

```
template<typename T>
void testing::internal::ThreadLocal< T >::set (
    const T & value) [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.106 Dokumentacja szablonu klasy std::tr1::tuple<>

```
#include <gtest-tuple.h>
```

Metody publiczne

- `tuple ()`
- `tuple (GTEST_BY_REF_(T0) f0, GTEST_BY_REF_(T1) f1, GTEST_BY_REF_(T2) f2, GTEST_BY_REF_(T3) f3, GTEST_BY_REF_(T4) f4, GTEST_BY_REF_(T5) f5, GTEST_BY_REF_(T6) f6, GTEST_BY_REF_(T7) f7, GTEST_BY_REF_(T8) f8, GTEST_BY_REF_(T9) f9)`
- `tuple (const tuple &t)`
- template<`GTEST_10_TYPENAMES_(U)`>
`tuple (const GTEST_10_TUPLE_(U)&t)`
- `tuple & operator=(const tuple &t)`
- template<`GTEST_10_TYPENAMES_(U)`>
`tuple & operator=(const GTEST_10_TUPLE_(U)&t)`
- template<`GTEST_10_TYPENAMES_(U)`>
`GTEST_DECLARE_TUPLE_AS_FRIEND_ tuple & CopyFrom (const GTEST_10_TUPLE_(U)&t)`

Atrybuty publiczne

- T0 `f0_`
- T1 `f1_`
- T2 `f2_`
- T3 `f3_`
- T4 `f4_`
- T5 `f5_`
- T6 `f6_`
- T7 `f7_`
- T8 `f8_`
- T9 `f9_`

Przyjaciele

- template<int k>
class `gtest_internal::Get`

8.106.1 Dokumentacja konstruktora i destruktora

8.106.1.1 `tuple()` [1/4]

```
template<GTEST_10_TYPENAMES_(T)>
std::tr1::tuple<>::tuple () [inline]
```

8.106.1.2 tuple() [2/4]

```
template<GTEST_10_TYPENAMES_(T)>
std::tr1::tuple<>::tuple (
    GTEST_BY_REF_(T0) f0,
    GTEST_BY_REF_(T1) f1,
    GTEST_BY_REF_(T2) f2,
    GTEST_BY_REF_(T3) f3,
    GTEST_BY_REF_(T4) f4,
    GTEST_BY_REF_(T5) f5,
    GTEST_BY_REF_(T6) f6,
    GTEST_BY_REF_(T7) f7,
    GTEST_BY_REF_(T8) f8,
    GTEST_BY_REF_(T9) f9) [inline], [explicit]
```

8.106.1.3 tuple() [3/4]

```
template<GTEST_10_TYPENAMES_(T)>
std::tr1::tuple<>::tuple (
    const tuple<> & t) [inline]
```

8.106.1.4 tuple() [4/4]

```
template<GTEST_10_TYPENAMES_(T)>
template<GTEST_10_TYPENAMES_(U)>
std::tr1::tuple<>::tuple (
    const GTEST_10_TUPLE_(U) & t) [inline]
```

8.106.2 Dokumentacja funkcji składowych

8.106.2.1 CopyFrom()

```
template<GTEST_10_TYPENAMES_(T)>
template<GTEST_10_TYPENAMES_(U)>
GTEST_DECLARE_TUPLE_AS_FRIEND_ tuple & std::tr1::tuple<>::CopyFrom (
    const GTEST_10_TUPLE_(U) & t) [inline]
```

8.106.2.2 operator=() [1/2]

```
template<GTEST_10_TYPENAMES_(T)>
template<GTEST_10_TYPENAMES_(U)>
tuple & std::tr1::tuple<>::operator= (
    const GTEST_10_TUPLE_(U) & t) [inline]
```

8.106.2.3 operator=() [2/2]

```
template<GTEST_10_TYPENAMES_(T)>
tuple & std::tr1::tuple<>::operator= (
    const tuple<> & t) [inline]
```

8.106.3 Dokumentacja przyjaciół i powiązanych symboli

8.106.3.1 gtest_internal::Get

```
template<GTEST_10_TYPENAMES_(T)>
template<int k>
friend class gtest_internal::Get [friend]
```

8.106.4 Dokumentacja atrybutów składowych

8.106.4.1 f0_

```
template<GTEST_10_TYPENAMES_(T)>
T0 std::tr1::tuple<>::f0_
```

8.106.4.2 f1_

```
template<GTEST_10_TYPENAMES_(T)>
T1 std::tr1::tuple<>::f1_
```

8.106.4.3 f2_

```
template<GTEST_10_TYPENAMES_(T)>
T2 std::tr1::tuple<>::f2_
```

8.106.4.4 f3_

```
template<GTEST_10_TYPENAMES_(T)>
T3 std::tr1::tuple<>::f3_
```

8.106.4.5 f4_

```
template<GTEST_10_TYPENAMES_(T)>
T4 std::tr1::tuple<>::f4_
```

8.106.4.6 f5_

```
template<GTEST_10_TYPENAMES_(T)>
T5 std::tr1::tuple<>::f5_
```

8.106.4.7 f6_

```
template<GTEST_10_TYPENAMES_(T)>
T6 std::tr1::tuple<>::f6_
```

8.106.4.8 f7_

```
template<GTEST_10_TYPENAMES_(T)>
T7 std::tr1::tuple<>::f7_
```

8.106.4.9 f8_

```
template<GTEST_10_TYPENAMES_(T)>
T8 std::tr1::tuple<>::f8_
```

8.106.4.10 f9_

```
template<GTEST_10_TYPENAMES_(T)>
T9 std::tr1::tuple<>::f9_
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.107 Dokumentacja klasy std::tr1::tuple<>

```
#include <gtest-tuple.h>
```

Metody publiczne

- `tuple ()`
- `tuple (const tuple &)`
- `tuple & operator= (const tuple &)`
- `GTEST_DECLARE_TUPLE_AS_FRIEND_tuple & CopyFrom (const GTEST_10_TUPLE_(U)&t)`

Atrybuty publiczne

- T0 `f0_`
- T1 `f1_`
- T2 `f2_`
- T3 `f3_`
- T4 `f4_`
- T5 `f5_`
- T6 `f6_`
- T7 `f7_`
- T8 `f8_`
- T9 `f9_`

Przyjaciele

- class `gtest_internal::Get`

8.107.1 Dokumentacja konstruktora i destruktora

8.107.1.1 tuple() [1/2]

```
std::tr1::tuple<>::tuple () [inline]
```

8.107.1.2 tuple() [2/2]

```
std::tr1::tuple<>::tuple (
    const tuple<> & ) [inline]
```

8.107.2 Dokumentacja funkcji składowych

8.107.2.1 CopyFrom()

```
GTEST_DECLARE_TUPLE_AS_FRIEND_ tuple & std::tr1::tuple<>::CopyFrom (
    const GTEST_10_TUPLE_(U) & t) [inline]
```

8.107.2.2 operator=(*)

```
tuple & std::tr1::tuple<>::operator= (
    const tuple<> & ) [inline]
```

8.107.3 Dokumentacja przyjaciół i powiązanych symboli

8.107.3.1 gtest_internal::Get

```
friend class gtest_internal::Get [friend]
```

8.107.4 Dokumentacja atrybutów składowych

8.107.4.1 f0_

```
T0 std::tr1::tuple<>::f0_
```

8.107.4.2 f1_

```
T1 std::tr1::tuple<>::f1_
```

8.107.4.3 f2_

```
T2 std::tr1::tuple<>::f2_
```

8.107.4.4 f3_

```
T3 std::tr1::tuple<>::f3_
```

8.107.4.5 f4_

```
T4 std::tr1::tuple<>::f4_
```

8.107.4.6 f5_

```
T5 std::tr1::tuple<>::f5_
```

8.107.4.7 f6_

```
T6 std::tr1::tuple<>::f6_
```

8.107.4.8 f7_

```
T7 std::tr1::tuple<>::f7_
```

8.107.4.9 f8_

```
T8 std::tr1::tuple<>::f8_
```

8.107.4.10 f9_

```
T9 std::tr1::tuple<>::f9_
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.108 Dokumentacja szablonu struktury std::tr1::tuple_element< k, Tuple >

```
#include <gtest-tuple.h>
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.109 Dokumentacja szablonu struktury std::tr1::tuple_size< Tuple >

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.110 Dokumentacja szablonu struktury std::tr1::tuple_size< GTEST_0_TUPLE_(T) >

```
#include <gtest-tuple.h>
```

Statyczne atrybuty publiczne

- static const int **value** = 0

8.110.1 Dokumentacja atrybutów składowych

8.110.1.1 **value**

```
template<GTEST_0_TYPENAMES_(T)>
const int std::tr1::tuple_size< GTEST_0_TUPLE_(T) >::value = 0 [static]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.111 Dokumentacja szablonu struktury std::tr1::tuple_size< GTEST_10_TUPLE_(T) >

```
#include <gtest-tuple.h>
```

Statyczne atrybuty publiczne

- static const int **value** = 10

8.111.1 Dokumentacja atrybutów składowych

8.111.1.1 **value**

```
template<GTEST_10_TYPENAMES_(T)>
const int std::tr1::tuple_size< GTEST_10_TUPLE_(T) >::value = 10 [static]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.112 Dokumentacja szablonu struktury std::tr1::tuple_size< GTEST_1_TUPLE_(T) >

```
#include <gtest-tuple.h>
```

Statyczne atrybuty publiczne

- static const int **value** = 1

8.112.1 Dokumentacja atrybutów składowych

8.112.1.1 **value**

```
template<GTEST_1_TYPENAMES_(T)>
const int std::tr1::tuple_size< GTEST_1_TUPLE_(T) >::value = 1 [static]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.113 Dokumentacja szablonu struktury std::tr1::tuple_size< GTEST_2_TUPLE_(T) >

```
#include <gtest-tuple.h>
```

Statyczne atrybuty publiczne

- static const int **value** = 2

8.113.1 Dokumentacja atrybutów składowych

8.113.1.1 **value**

```
template<GTEST_2_TYPENAMES_(T)>
const int std::tr1::tuple_size< GTEST_2_TUPLE_(T) >::value = 2 [static]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.114 Dokumentacja szablonu struktury std::tr1::tuple_size< GTEST_3_TUPLE_(T) >

```
#include <gtest-tuple.h>
```

Statyczne atrybuty publiczne

- static const int **value** = 3

8.114.1 Dokumentacja atrybutów składowych

8.114.1.1 **value**

```
template<GTEST_3_TYPENAMES_(T)>
const int std::tr1::tuple_size< GTEST_3_TUPLE_(T) >::value = 3 [static]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.115 Dokumentacja szablonu struktury std::tr1::tuple_size< GTEST_4_TUPLE_(T) >

```
#include <gtest-tuple.h>
```

Statyczne atrybuty publiczne

- static const int **value** = 4

8.115.1 Dokumentacja atrybutów składowych

8.115.1.1 **value**

```
template<GTEST_4_TYPENAMES_(T)>
const int std::tr1::tuple_size< GTEST_4_TUPLE_(T) >::value = 4 [static]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.116 Dokumentacja szablonu struktury std::tr1::tuple_size< GTEST_5_TUPLE_(T) >

```
#include <gtest-tuple.h>
```

Statyczne atrybuty publiczne

- static const int **value** = 5

8.116.1 Dokumentacja atrybutów składowych

8.116.1.1 **value**

```
template<GTEST_5_TYPENAMES_(T)>
const int std::tr1::tuple_size< GTEST_5_TUPLE_(T) >::value = 5 [static]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.117 Dokumentacja szablonu struktury std::tr1::tuple_size< GTEST_6_TUPLE_(T) >

```
#include <gtest-tuple.h>
```

Statyczne atrybuty publiczne

- static const int **value** = 6

8.117.1 Dokumentacja atrybutów składowych

8.117.1.1 **value**

```
template<GTEST_6_TYPENAMES_(T)>
const int std::tr1::tuple_size< GTEST_6_TUPLE_(T) >::value = 6 [static]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.118 Dokumentacja szablonu struktury std::tr1::tuple_size< GTEST_7_TUPLE_(T) >

```
#include <gtest-tuple.h>
```

Statyczne atrybuty publiczne

- static const int **value** = 7

8.118.1 Dokumentacja atrybutów składowych

8.118.1.1 **value**

```
template<GTEST_7_TYPENAMES_(T)>
const int std::tr1::tuple_size< GTEST_7_TUPLE_(T) >::value = 7 [static]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.119 Dokumentacja szablonu struktury std::tr1::tuple_size< GTEST_8_TUPLE_(T) >

```
#include <gtest-tuple.h>
```

Statyczne atrybuty publiczne

- static const int **value** = 8

8.119.1 Dokumentacja atrybutów składowych

8.119.1.1 **value**

```
template<GTEST_8_TYPENAMES_(T)>
const int std::tr1::tuple_size< GTEST_8_TUPLE_(T) >::value = 8 [static]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.120 Dokumentacja szablonu struktury `std::tr1::tuple_size< GTEST_9_TUPLE_(T) >`

```
#include <gtest-tuple.h>
```

Statyczne atrybuty publiczne

- static const int `value` = 9

8.120.1 Dokumentacja atrybutów składowych

8.120.1.1 `value`

```
template<GTEST_9_TYPENAMES_(T)>
const int std::tr1::tuple_size< GTEST_9_TUPLE_(T) >::value = 9 [static]
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.121 Dokumentacja szablonu struktury `std::tr1::gtest_internal::TupleElement< kIndexValid, kIndex, Tuple >`

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.122 Dokumentacja szablonu struktury `std::tr1::gtest_internal::TupleElement< true, 0, GTEST_10_TUPLE_(T) >`

```
#include <gtest-tuple.h>
```

Typy publiczne

- typedef T0 `type`

8.122.1 Dokumentacja składowych definicji typu

8.122.1.1 type

```
template<GTEST_10_TYPENAMES_(T)>
typedef T0 std::tr1::gtest_internal::TupleElement< true, 0, GTEST_10_TUPLE_(T) >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.123 Dokumentacja szablonu struktury

`std::tr1::gtest_internal::TupleElement< true, 1, GTEST_10_TUPLE_(T) >`

```
#include <gtest-tuple.h>
```

Typy publiczne

- `typedef T1 type`

8.123.1 Dokumentacja składowych definicji typu

8.123.1.1 type

```
template<GTEST_10_TYPENAMES_(T)>
typedef T1 std::tr1::gtest_internal::TupleElement< true, 1, GTEST_10_TUPLE_(T) >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.124 Dokumentacja szablonu struktury

`std::tr1::gtest_internal::TupleElement< true, 2, GTEST_10_TUPLE_(T) >`

```
#include <gtest-tuple.h>
```

Typy publiczne

- `typedef T2 type`

8.124.1 Dokumentacja składowych definicji typu

8.124.1.1 type

```
template<GTEST_10_TYPENAMES_(T)>
typedef T2 std::tr1::gtest_internal::TupleElement< true, 2, GTEST_10_TUPLE_(T) >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.125 Dokumentacja szablonu struktury

```
std::tr1::gtest_internal::TupleElement< true, 3,
GTEST_10_TUPLE_(T) >
```

```
#include <gtest-tuple.h>
```

Typy publiczne

- typedef T3 type

8.125.1 Dokumentacja składowych definicji typu

8.125.1.1 type

```
template<GTEST_10_TYPENAMES_(T)>
typedef T3 std::tr1::gtest_internal::TupleElement< true, 3, GTEST_10_TUPLE_(T) >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.126 Dokumentacja szablonu struktury

```
std::tr1::gtest_internal::TupleElement< true, 4,
GTEST_10_TUPLE_(T) >
```

```
#include <gtest-tuple.h>
```

Typy publiczne

- typedef T4 type

8.126.1 Dokumentacja składowych definicji typu

8.126.1.1 type

```
template<GTEST_10_TYPENAMES_(T)>
typedef T4 std::tr1::gtest_internal::TupleElement< true, 4, GTEST_10_TUPLE_(T) >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.127 Dokumentacja szablonu struktury

`std::tr1::gtest_internal::TupleElement< true, 5, GTEST_10_TUPLE_(T) >`

```
#include <gtest-tuple.h>
```

Typy publiczne

- `typedef T5 type`

8.127.1 Dokumentacja składowych definicji typu

8.127.1.1 type

```
template<GTEST_10_TYPENAMES_(T)>
typedef T5 std::tr1::gtest_internal::TupleElement< true, 5, GTEST_10_TUPLE_(T) >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.128 Dokumentacja szablonu struktury

`std::tr1::gtest_internal::TupleElement< true, 6, GTEST_10_TUPLE_(T) >`

```
#include <gtest-tuple.h>
```

Typy publiczne

- `typedef T6 type`

8.128.1 Dokumentacja składowych definicji typu

8.128.1.1 type

```
template<GTEST_10_TYPENAMES_(T)>
typedef T6 std::tr1::gtest_internal::TupleElement< true, 6, GTEST_10_TUPLE_(T) >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.129 Dokumentacja szablonu struktury

```
std::tr1::gtest_internal::TupleElement< true, 7,
GTEST_10_TUPLE_(T) >
```

```
#include <gtest-tuple.h>
```

Typy publiczne

- typedef T7 type

8.129.1 Dokumentacja składowych definicji typu

8.129.1.1 type

```
template<GTEST_10_TYPENAMES_(T)>
typedef T7 std::tr1::gtest_internal::TupleElement< true, 7, GTEST_10_TUPLE_(T) >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.130 Dokumentacja szablonu struktury

```
std::tr1::gtest_internal::TupleElement< true, 8,
GTEST_10_TUPLE_(T) >
```

```
#include <gtest-tuple.h>
```

Typy publiczne

- typedef T8 type

8.130.1 Dokumentacja składowych definicji typu

8.130.1.1 type

```
template<GTEST_10_TYPENAMES_(T)>
typedef T8 std::tr1::gtest_internal::TupleElement< true, 8, GTEST_10_TUPLE_(T) >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.131 Dokumentacja szablonu struktury

std::tr1::gtest_internal::TupleElement< true, 9, GTEST_10_TUPLE_(T) >

```
#include <gtest-tuple.h>
```

Typy publiczne

- typedef T9 type

8.131.1 Dokumentacja składowych definicji typu

8.131.1.1 type

```
template<GTEST_10_TYPENAMES_(T)>
typedef T9 std::tr1::gtest_internal::TupleElement< true, 9, GTEST_10_TUPLE_(T) >::type
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

8.132 Dokumentacja szablonu struktury testing::internal::TuplePolicy< TupleT >

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

8.133 Dokumentacja szablonu klasy testing::internal::TypeHelper< T >

```
#include <gtest-internal.h>
```

Statyczne atrybuty publiczne

- static bool [dummy_](#)

8.133.1 Dokumentacja atrybutów składowych**8.133.1.1 dummy_**

```
template<typename T>
bool testing::internal::TypeIdHelper< T >::dummy_ [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-internal.h](#)

8.134 Dokumentacja szablonu klasy**testing::internal2::TypeWithoutFormatter< T, kTypeKind >**

```
#include <gtest-printers.h>
```

Statyczne metody publiczne

- static void [PrintValue](#) (const T &value, ::std::ostream *os)

8.134.1 Dokumentacja funkcji składowych**8.134.1.1 PrintValue()**

```
template<typename T, TypeKind kTypeKind>
void testing::internal2::TypeWithoutFormatter< T, kTypeKind >::PrintValue (
    const T & value,
    ::std::ostream * os) [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/[gtest-printers.h](#)

8.135 Dokumentacja szablonu klasy**testing::internal2::TypeWithoutFormatter< T,
kConvertibleToInteger >**

```
#include <gtest-printers.h>
```

Statyczne metody publiczne

- static void [PrintValue](#) (const T &value, ::std::ostream *os)

8.135.1 Dokumentacja funkcji składowych

8.135.1.1 PrintValue()

```
template<typename T>
void testing::internal2::TypeWithoutFormatter< T, kConvertibleToInteger >::PrintValue (
    const T & value,
    ::std::ostream * os) [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

8.136 Dokumentacja szablonu klasy testing::internal2::TypeWithoutFormatter< T, kProtobuf >

```
#include <gtest-printers.h>
```

Statyczne metody publiczne

- static void [PrintValue](#) (const T &value, ::std::ostream *os)

8.136.1 Dokumentacja funkcji składowych

8.136.1.1 PrintValue()

```
template<typename T>
void testing::internal2::TypeWithoutFormatter< T, kProtobuf >::PrintValue (
    const T & value,
    ::std::ostream * os) [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

8.137 Dokumentacja szablonu klasy testing::internal::TypeWithSize< size >

```
#include <gtest-port.h>
```

Typy publiczne

- `typedef void UInt`

8.137.1 Dokumentacja składowych definicji typu**8.137.1.1 UInt**

```
template<size_t size>
typedef void testing::internal::TypeWithSize< size >::UInt
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.138 Dokumentacja klasy testing::internal::TypeWithSize< 4 >

```
#include <gtest-port.h>
```

Typy publiczne

- `typedef int Int`
- `typedef unsigned int UInt`

8.138.1 Dokumentacja składowych definicji typu**8.138.1.1 Int**

```
typedef int testing::internal::TypeWithSize< 4 >::Int
```

8.138.1.2 UInt

```
typedef unsigned int testing::internal::TypeWithSize< 4 >::UInt
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

8.139 Dokumentacja klasy testing::internal::TypeWithSize< 8 >

```
#include <gtest-port.h>
```

Typy publiczne

- `typedef long long Int`
- `typedef unsigned long long UInt`

8.139.1 Dokumentacja składowych definicji typu

8.139.1.1 Int

```
typedef long long testing::internal::TypeWithSize< 8 >::Int
```

8.139.1.2 UInt

```
typedef unsigned long long testing::internal::TypeWithSize< 8 >::UInt
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- `packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h`

8.140 Dokumentacja klasy testing::UnitTest

```
#include <gtest.h>
```

Metody publiczne

- `int Run () GTEST_MUST_USE_RESULT_`
- `const char * original_working_dir () const`
- `const TestCase * current_test_case () const GTEST_LOCK_EXCLUDED_(mutex_)`
- `const TestInfo * current_test_info () const GTEST_LOCK_EXCLUDED_(mutex_)`
- `int random_seed () const`
- `internal::ParameterizedTestCaseRegistry & parameterized_test_registry () GTEST_LOCK_EXCLUDED_(mutex_)`
- `int successful_test_case_count () const`
- `int failed_test_case_count () const`
- `int total_test_case_count () const`
- `int test_case_to_run_count () const`
- `int successful_test_count () const`
- `int failed_test_count () const`
- `int reportable_disabled_test_count () const`
- `int disabled_test_count () const`
- `int reportable_test_count () const`
- `int total_test_count () const`
- `int test_to_run_count () const`
- `TimeInMillis start_timestamp () const`
- `TimeInMillis elapsed_time () const`
- `bool Passed () const`
- `bool Failed () const`
- `const TestCase * GetTestCase (int i) const`
- `const TestResult & ad_hoc_test_result () const`
- `TestEventListeners & listeners ()`

Statyczne metody publiczne

- static UnitTest * [GetInstance \(\)](#)

Przyjaciele

- class [ScopedTrace](#)
- class [Test](#)
- class [internal::AssertHelper](#)
- class [internal::StreamingListenerTest](#)
- class [internal::UnitTestRecordPropertyTestHelper](#)
- [Environment * AddGlobalTestEnvironment \(Environment *env\)](#)
- [internal::UnitTestImpl * internal::GetUnitTestImpl \(\)](#)
- void [internal::ReportFailureInUnknownLocation \(TestPartResult::Type result_type, const std::string &message\)](#)

8.140.1 Dokumentacja funkcji składowych

8.140.1.1 ad_hoc_test_result()

```
const TestResult & testing::UnitTest::ad_hoc_test_result () const
```

8.140.1.2 current_test_case()

```
const TestCase * testing::UnitTest::current_test_case () const
```

8.140.1.3 current_test_info()

```
const TestInfo * testing::UnitTest::current_test_info () const
```

8.140.1.4 disabled_test_count()

```
int testing::UnitTest::disabled_test_count () const
```

8.140.1.5 elapsed_time()

```
TimeInMillis testing::UnitTest::elapsed_time () const
```

8.140.1.6 Failed()

```
bool testing::UnitTest::Failed () const
```

8.140.1.7 failed_test_case_count()

```
int testing::UnitTest::failed_test_case_count () const
```

8.140.1.8 failed_test_count()

```
int testing::UnitTest::failed_test_count () const
```

8.140.1.9 GetInstance()

```
UnitTest * testing::UnitTest::GetInstance () [static]
```

8.140.1.10 GetTestCase()

```
const TestCase * testing::UnitTest::GetTestCase (
    int i) const
```

8.140.1.11 listeners()

```
TestEventListeners & testing::UnitTest::listeners ()
```

8.140.1.12 original_working_dir()

```
const char * testing::UnitTest::original_working_dir () const
```

8.140.1.13 parameterized_test_registry()

```
internal::ParameterizedTestCaseRegistry & testing::UnitTest::parameterized_test_registry ()
```

8.140.1.14 Passed()

```
bool testing::UnitTest::Passed () const
```

8.140.1.15 random_seed()

```
int testing::UnitTest::random_seed () const
```

8.140.1.16 reportable_disabled_test_count()

```
int testing::UnitTest::reportable_disabled_test_count () const
```

8.140.1.17 reportable_test_count()

```
int testing::UnitTest::reportable_test_count () const
```

8.140.1.18 Run()

```
int testing::UnitTest::Run ()
```

8.140.1.19 start_timestamp()

```
TimeInMillis testing::UnitTest::start_timestamp () const
```

8.140.1.20 successful_test_case_count()

```
int testing::UnitTest::successful_test_case_count () const
```

8.140.1.21 successful_test_count()

```
int testing::UnitTest::successful_test_count () const
```

8.140.1.22 test_case_to_run_count()

```
int testing::UnitTest::test_case_to_run_count () const
```

8.140.1.23 test_to_run_count()

```
int testing::UnitTest::test_to_run_count () const
```

8.140.1.24 total_test_case_count()

```
int testing::UnitTest::total_test_case_count () const
```

8.140.1.25 total_test_count()

```
int testing::UnitTest::total_test_count () const
```

8.140.2 Dokumentacja przyjaciół i powiązanych symboli**8.140.2.1 AddGlobalTestEnvironment**

```
Environment * AddGlobalTestEnvironment (
    Environment * env) [friend]
```

8.140.2.2 internal::AssertHelper

```
friend class internal::AssertHelper [friend]
```

8.140.2.3 internal::GetUnitTestImpl

```
internal::UnitTestImpl * internal::GetUnitTestImpl () [friend]
```

8.140.2.4 internal::ReportFailureInUnknownLocation

```
void internal::ReportFailureInUnknownLocation (
    TestPartResult::Type result_type,
    const std::string & message) [friend]
```

8.140.2.5 internal::StreamingListenerTest

```
friend class internal::StreamingListenerTest [friend]
```

8.140.2.6 internal::UnitTestRecordPropertyTestHelper

```
friend class internal::UnitTestRecordPropertyTestHelper [friend]
```

8.140.2.7 ScopedTrace

```
friend class ScopedTrace [friend]
```

8.140.2.8 Test

```
friend class Test [friend]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

8.141 Dokumentacja szablonu klasy testing::internal::UniversalPrinter< T >

```
#include <gtest-printers.h>
```

Statyczne metody publiczne

- static void [Print](#) (const T &[value](#), ::std::ostream *[os](#))

8.141.1 Dokumentacja funkcji składowych

8.141.1.1 Print()

```
template<typename T>
void testing::internal::UniversalPrinter< T >::Print (
    const T & value,
    ::std::ostream * os) [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

8.142 Dokumentacja szablonu klasy testing::internal::UniversalPrinter< T & >

```
#include <gtest-printers.h>
```

Statyczne metody publiczne

- static void Print (const T &**value**, ::std::ostream *os)

8.142.1 Dokumentacja funkcji składowych

8.142.1.1 Print()

```
template<typename T>
void testing::internal::UniversalPrinter< T & >::Print (
    const T & value,
    ::std::ostream * os) [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

8.143 Dokumentacja szablonu klasy testing::internal::UniversalPrinter< T[N]>

```
#include <gtest-printers.h>
```

Statyczne metody publiczne

- static void Print (const T(&a)[N], ::std::ostream *os)

8.143.1 Dokumentacja funkcji składowych

8.143.1.1 Print()

```
template<typename T, size_t N>
void testing::internal::UniversalPrinter< T[N]>::Print (
    const T(&) a[N],
    ::std::ostream * os) [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

8.144 Dokumentacja szablonu klasy testing::internal::UniversalTersePrinter< T >

```
#include <gtest-printers.h>
```

Statyczne metody publiczne

- static void Print (const T &**value**, ::std::ostream *os)

8.144.1 Dokumentacja funkcji składowych

8.144.1.1 Print()

```
template<typename T>
void testing::internal::UniversalTersePrinter< T >::Print (
    const T & value,
    ::std::ostream * os) [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

8.145 Dokumentacja klasy testing::internal::UniversalTersePrinter< char * >

```
#include <gtest-printers.h>
```

Statyczne metody publiczne

- static void Print (char *str, ::std::ostream *os)

8.145.1 Dokumentacja funkcji składowych

8.145.1.1 Print()

```
void testing::internal::UniversalTersePrinter< char * >::Print (
    char * str,
    ::std::ostream * os) [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

8.146 Dokumentacja klasy testing::internal::UniversalTersePrinter< const char * >

```
#include <gtest-printers.h>
```

Statyczne metody publiczne

- static void [Print](#) (const char *str, ::std::ostream *os)

8.146.1 Dokumentacja funkcji składowych

8.146.1.1 Print()

```
void testing::internal::UniversalTersePrinter< const char * >::Print (
    const char * str,
    ::std::ostream * os) [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

8.147 Dokumentacja szablonu klasy testing::internal::UniversalTersePrinter< T & >

```
#include <gtest-printers.h>
```

Statyczne metody publiczne

- static void [Print](#) (const T &[value](#), ::std::ostream *os)

8.147.1 Dokumentacja funkcji składowych

8.147.1.1 Print()

```
template<typename T>
void testing::internal::UniversalTersePrinter< T & >::Print (
    const T & value,
    ::std::ostream * os) [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

8.148 Dokumentacja szablonu klasy testing::internal::UniversalTersePrinter< T[N]>

```
#include <gtest-printers.h>
```

Statyczne metody publiczne

- static void Print (const T(&value)[N], ::std::ostream *os)

8.148.1 Dokumentacja funkcji składowych

8.148.1.1 Print()

```
template<typename T, size_t N>
void testing::internal::UniversalTersePrinter< T[N]>::Print (
    const T(&) value[N],
    ::std::ostream * os) [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

8.149 Dokumentacja klasy testing::internal::UniversalTersePrinter< wchar_t *>

```
#include <gtest-printers.h>
```

Statyczne metody publiczne

- static void Print (wchar_t *str, ::std::ostream *os)

8.149.1 Dokumentacja funkcji składowych

8.149.1.1 Print()

```
void testing::internal::UniversalTersePrinter< wchar_t * >::Print (
    wchar_t * str,
    ::std::ostream * os)  [inline], [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

8.150 Dokumentacja szablonu klasy testing::internal::ValueArray1< T1 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- **ValueArray1** (T1 v1)
- template<typename T>
operator **ParamGenerator< T >** () const
- **ValueArray1** (const ValueArray1 &other)

8.150.1 Dokumentacja konstruktora i destruktora

8.150.1.1 ValueArray1() [1/2]

```
template<typename T1>
testing::internal::ValueArray1< T1 >::ValueArray1 (
    T1 v1)  [inline], [explicit]
```

8.150.1.2 ValueArray1() [2/2]

```
template<typename T1>
testing::internal::ValueArray1< T1 >::ValueArray1 (
    const ValueArray1< T1 > & other)  [inline]
```

8.150.2 Dokumentacja funkcji składowych

8.150.2.1 operator ParamGenerator< T >()

```
template<typename T1>
template<typename T>
testing::internal::ValueArray1< T1 >::operator ParamGenerator< T > () const  [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

8.151 Dokumentacja szablonu klasy testing::internal::ValueArray10< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray10` (`T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10`)
- `template<typename T>`
`operator ParamGenerator< T > () const`
- `ValueArray10` (`const ValueArray10 &other`)

8.151.1 Dokumentacja konstruktora i destruktora

8.151.1.1 `ValueArray10()` [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10>
testing::internal::ValueArray10< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 >::ValueArray10 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10)  [inline]
```

8.151.1.2 `ValueArray10()` [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10>
testing::internal::ValueArray10< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 >::ValueArray10 (
    const ValueArray10< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 > & other)  [inline]
```

8.151.2 Dokumentacja funkcji składowych

8.151.2.1 `operator ParamGenerator< T >()`

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10>
template<typename T>
testing::internal::ValueArray10< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 >::operator ParamGenerator< T > () const  [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-generated.h

8.152 Dokumentacja szablonu klasy testing::internal::ValueArray11< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray11` (`T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11`)
- `template<typename T> operator ParamGenerator< T > () const`
- `ValueArray11 (const ValueArray11 &other)`

8.152.1 Dokumentacja konstruktora i destruktora

8.152.1.1 ValueArray11() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
typename T8, typename T9, typename T10, typename T11>
testing::internal::ValueArray11< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11 >::ValueArray11
(
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11) [inline]
```

8.152.1.2 ValueArray11() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
typename T8, typename T9, typename T10, typename T11>
testing::internal::ValueArray11< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11 >::ValueArray11
(
    const ValueArray11< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11 > & other) [inline]
```

8.152.2 Dokumentacja funkcji składowych

8.152.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
typename T8, typename T9, typename T10, typename T11>
template<typename T>
testing::internal::ValueArray11< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11 >::operator
ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-generated.h

8.153 Dokumentacja szablonu klasy testing::internal::ValueArray12< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray12` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12)
- template<typename T>
 `operator ParamGenerator< T >` () const
- `ValueArray12` (const ValueArray12 &other)

8.153.1 Dokumentacja konstruktora i destruktora

8.153.1.1 ValueArray12() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12>
testing::internal::ValueArray12< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12 >::Value<-
Array12 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12)  [inline]
```

8.153.1.2 ValueArray12() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12>
testing::internal::ValueArray12< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12 >::Value<-
Array12 (
    const ValueArray12< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12 > & other)
[inline]
```

8.153.2 Dokumentacja funkcji składowych

8.153.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12>
template<typename T>
testing::internal::ValueArray12< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12 >::operator
ParamGenerator< T > () const  [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

8.154 Dokumentacja szablonu klasy testing::internal::ValueArray13< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray13` (`T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13`)
- template<typename T>
 `operator ParamGenerator< T > () const`
- `ValueArray13` (`const ValueArray13 &other`)

8.154.1 Dokumentacja konstruktora i destruktora

8.154.1.1 ValueArray13() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13>
testing::internal::ValueArray13< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13 >::←
ValueArray13 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13) [inline]
```

8.154.1.2 ValueArray13() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13>
testing::internal::ValueArray13< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13 >::←
ValueArray13 (
    const ValueArray13< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13 > &
other) [inline]
```

8.154.2 Dokumentacja funkcji składowych

8.154.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13>
template<typename T>
testing::internal::ValueArray13< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13 >↔
::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.155 Dokumentacja szablonu klasy testing::internal::ValueArray14< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray14` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14)
- template<typename T>
`operator ParamGenerator< T > () const`
- `ValueArray14` (const ValueArray14 &other)

8.155.1 Dokumentacja konstruktora i destruktora

8.155.1.1 ValueArray14() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14>
testing::internal::ValueArray14< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14 >::ValueArray14 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14) [inline]
```

8.155.1.2 `ValueArray14()` [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14>
testing::internal::ValueArray14< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14
>::ValueArray14 (
    const ValueArray14< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14 >
& other) [inline]
```

8.155.2 Dokumentacja funkcji składowych

8.155.2.1 `operator ParamGenerator< T >()`

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14>
template<typename T>
testing::internal::ValueArray14< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14
>::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.156 Dokumentacja szablonu klasy `testing::internal::ValueArray15< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15 >`

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray15` (`T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15`)
- `template<typename T> operator ParamGenerator< T > () const`
- `ValueArray15 (const ValueArray15 &other)`

8.156.1 Dokumentacja konstruktora i destruktora

8.156.1.1 ValueArray15() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15>
testing::internal::ValueArray15< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15 >::ValueArray15 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15)  [inline]
```

8.156.1.2 ValueArray15() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15>
testing::internal::ValueArray15< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15 >::ValueArray15 (
    const ValueArray15< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15 > & other)  [inline]
```

8.156.2 Dokumentacja funkcji składowych

8.156.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15>
template<typename T>
testing::internal::ValueArray15< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15 >::operator ParamGenerator< T > () const  [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.157 Dokumentacja szablonu klasy testing::internal::ValueArray16< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- [ValueArray16](#) (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16)
- template<typename T>
[operator ParamGenerator< T >](#) () const
- [ValueArray16](#) (const ValueArray16 &other)

8.157.1 Dokumentacja konstruktora i destruktora

8.157.1.1 ValueArray16() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14,
typename T15, typename T16>::ValueArray16 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16) [inline]
```

8.157.1.2 ValueArray16() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14,
typename T15, typename T16>::ValueArray16 (
    const ValueArray16< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16 > & other) [inline]
```

8.157.2 Dokumentacja funkcji składowych

8.157.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16>
template<typename T>
testing::internal::ValueArray16< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-generated.h

8.158 Dokumentacja szablonu klasy testing::internal::ValueArray17< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray17` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17)
- template<typename T>
 `operator ParamGenerator< T > () const`
- `ValueArray17` (const ValueArray17 &other)

8.158.1 Dokumentacja konstruktora i destruktora

8.158.1.1 ValueArray17() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17>::ValueArray17 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17) [inline]
```

8.158.1.2 `ValueArray17()` [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17>
testing::internal::ValueArray17< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17 >::ValueArray17 (
    const ValueArray17< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17 > & other) [inline]
```

8.158.2 Dokumentacja funkcji składowych

8.158.2.1 `operator ParamGenerator< T >()`

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17>
template<typename T>
testing::internal::ValueArray17< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.159 Dokumentacja szablonu klasy `testing::internal::ValueArray18< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18 >`

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray18` (`T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18`)
- `template<typename T> operator ParamGenerator< T > () const`
- `ValueArray18` (`const ValueArray18 &other`)

8.159.1 Dokumentacja konstruktora i destruktora

8.159.1.1 ValueArray18() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18>
testing::internal::ValueArray18< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18 >::ValueArray18 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18)  [inline]
```

8.159.1.2 ValueArray18() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18>
testing::internal::ValueArray18< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18 >::ValueArray18 (
    const ValueArray18< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18 > & other)  [inline]
```

8.159.2 Dokumentacja funkcji składowych

8.159.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18>
template<typename T>
testing::internal::ValueArray18< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18 >::operator ParamGenerator< T > () const  [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.160 Dokumentacja szablonu klasy testing::internal::ValueArray19< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray19` (`T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19`)
- template<typename T>
 `operator ParamGenerator< T > () const`
- `ValueArray19` (`const ValueArray19 &other`)

8.160.1 Dokumentacja konstruktora i destruktora

8.160.1.1 ValueArray19() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19>
testing::internal::ValueArray19< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19 >::ValueArray19 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19)  [inline]
```

8.160.1.2 ValueArray19() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19>
testing::internal::ValueArray19< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19 >::ValueArray19 (
    const ValueArray19< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19 > & other)  [inline]
```

8.160.2 Dokumentacja funkcji składowych

8.160.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19>
template<typename T>
testing::internal::ValueArray19< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.161 Dokumentacja szablonu klasy testing::internal::ValueArray2< T1, T2 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray2` (T1 v1, T2 v2)
- `template<typename T> operator ParamGenerator< T > () const`
- `ValueArray2` (const `ValueArray2 &other`)

8.161.1 Dokumentacja konstruktora i destruktora

8.161.1.1 ValueArray2() [1/2]

```
template<typename T1, typename T2>
testing::internal::ValueArray2< T1, T2 >::ValueArray2 (
    T1 v1,
    T2 v2) [inline]
```

8.161.1.2 ValueArray2() [2/2]

```
template<typename T1, typename T2>
testing::internal::ValueArray2< T1, T2 >::ValueArray2 (
    const ValueArray2< T1, T2 > & other) [inline]
```

8.161.2 Dokumentacja funkcji składowych

8.161.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2>
template<typename T>
testing::internal::ValueArray2< T1, T2 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.162 Dokumentacja szablonu klasy testing::internal::ValueArray2< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray20` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20)
- template<typename T>
 operator ParamGenerator< T > () const
- `ValueArray20` (const ValueArray20 &other)

8.162.1 Dokumentacja konstruktora i destruktora

8.162.1.1 ValueArray20() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20>
testing::internal::ValueArray20< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20 >::ValueArray20 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20) [inline]
```

8.162.1.2 ValueArray20() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20>
testing::internal::ValueArray20< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20 >::ValueArray20
    const ValueArray20< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20 > & other) [inline]
```

8.162.2 Dokumentacja funkcji składowych

8.162.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20>
template<typename T>
testing::internal::ValueArray20< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.163 Dokumentacja szablonu klasy testing::internal::ValueArray21< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- **ValueArray21** (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21)
- **template<typename T>**
operator ParamGenerator< T > () const
- **ValueArray21** (const ValueArray21 &other)

8.163.1 Dokumentacja konstruktora i destruktora

8.163.1.1 ValueArray21() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21>
testing::internal::ValueArray21< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21 >::ValueArray21 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21) [inline]
```

8.163.1.2 ValueArray21() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21>
testing::internal::ValueArray21< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21 >::ValueArray21 (
    const ValueArray21< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21 > & other) [inline]
```

8.163.2 Dokumentacja funkcji składowych

8.163.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21>
template<typename T>
testing::internal::ValueArray21< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.164 Dokumentacja szablonu klasy testing::internal::ValueArray22< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray22` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22)
- template<typename T>
 `operator ParamGenerator< T >` () const
- `ValueArray22` (const ValueArray22 &other)

8.164.1 Dokumentacja konstruktora i destruktora

8.164.1.1 ValueArray22() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22>
testing::internal::ValueArray22< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22 >::ValueArray22 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22) [inline]
```

8.164.1.2 `ValueArray22()` [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22>
testing::internal::ValueArray22< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22 >::ValueArray22 (
    const ValueArray22< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22 > & other) [inline]
```

8.164.2 Dokumentacja funkcji składowych

8.164.2.1 operator `ParamGenerator< T >()`

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22>
template<typename T>
testing::internal::ValueArray22< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.165 Dokumentacja szablonu klasy `testing::internal::ValueArray23< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23 >`

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray23` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23)
- template<typename T>
operator `ParamGenerator< T >` () const
- `ValueArray23` (const `ValueArray23` &other)

8.165.1 Dokumentacja konstruktora i destruktora

8.165.1.1 ValueArray23() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23>
testing::internal::ValueArray23< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23 >::ValueArray23 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23) [inline]
```

8.165.1.2 ValueArray23() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23>
testing::internal::ValueArray23< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23 >::ValueArray23 (
    const ValueArray23< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23 > & other) [inline]
```

8.165.2 Dokumentacja funkcji składowych

8.165.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23>
```

```
template<typename T>
testing::internal::ValueArray23< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.166 Dokumentacja szablonu klasy testing::internal::ValueArray24< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray24` (`T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24`)
- `template<typename T> operator ParamGenerator< T > () const`
- `ValueArray24 (const ValueArray24 &other)`

8.166.1 Dokumentacja konstruktora i destruktora

8.166.1.1 ValueArray24() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24>
testing::internal::ValueArray24< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24 >::ValueArray24 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24) [inline]
```

8.166.1.2 ValueArray24() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24>
testing::internal::ValueArray24< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24 >::ValueArray24 (
    const ValueArray24< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24 > & other) [inline]
```

8.166.2 Dokumentacja funkcji składowych

8.166.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24>
template<typename T>
testing::internal::ValueArray24< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24 >::operator ParamGenerator< T > () const
[inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.167 Dokumentacja szablonu klasy testing::internal::ValueArray25< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- [ValueArray25](#) (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25)
- template<typename T>
 [operator ParamGenerator< T >\(\)](#) const
- [ValueArray25](#) (const ValueArray25 &other)

8.167.1 Dokumentacja konstruktora i destruktora

8.167.1.1 ValueArray25() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25>
testing::internal::ValueArray25< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25 >::ValueArray25 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25)  [inline]
```

8.167.1.2 ValueArray25() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25>
testing::internal::ValueArray25< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25 >::ValueArray25 (
    const ValueArray25< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25 > & other)  [inline]
```

8.167.2 Dokumentacja funkcji składowych

8.167.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
```

```
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25>  
template<typename T>  
testing::internal::ValueArray25< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,  
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25 >::operator ParamGenerator< T > ()  
const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.168 Dokumentacja szablonu klasy testing::internal::ValueArray26< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- **ValueArray26** (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26)
- template<typename T>
operator ParamGenerator< T > () const
- **ValueArray26** (const ValueArray26 &other)

8.168.1 Dokumentacja konstruktora i destruktora

8.168.1.1 ValueArray26() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26>  
testing::internal::ValueArray26< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,  
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26 >::ValueArray26 (  
    T1 v1,  
    T2 v2,  
    T3 v3,  
    T4 v4,  
    T5 v5,  
    T6 v6,  
    T7 v7,  
    T8 v8,  
    T9 v9,  
    T10 v10,  
    T11 v11,  
    T12 v12,  
    T13 v13,  
    T14 v14,
```

```

T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26) [inline]

```

8.168.1.2 ValueArray26() [2/2]

```

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26>
testing::internal::ValueArray26< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26 >::ValueArray26(
    const ValueArray26< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26 > & other) [inline]

```

8.168.2 Dokumentacja funkcji składowych

8.168.2.1 operator ParamGenerator< T >()

```

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26>
template<typename T>
testing::internal::ValueArray26< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26 >::operator ParamGenerator< T >()
const [inline]

```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.169 Dokumentacja szablonu klasy testing::internal::ValueArray27< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray27` (`T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27)`
- template<typename T>
`operator ParamGenerator< T > () const`
- `ValueArray27` (`const ValueArray27 &other`)

8.169.1 Dokumentacja konstruktora i destruktora

8.169.1.1 ValueArray27() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27>
testing::internal::ValueArray27< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27 >::ValueArray27 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27) [inline]
```

8.169.1.2 ValueArray27() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27>
testing::internal::ValueArray27< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27 >::ValueArray27 (
    const ValueArray27< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27 > & other) [inline]
```

8.169.2 Dokumentacja funkcji składowych

8.169.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27>
template<typename T>
testing::internal::ValueArray27< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27 >::operator ParamGenerator< T
> () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-generated.h

8.170 Dokumentacja szablonu klasy testing::internal::ValueArray28< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray28` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28)
- template<typename T>
operator ParamGenerator< T > () const
- `ValueArray28` (const ValueArray28 &other)

8.170.1 Dokumentacja konstruktora i destruktora

8.170.1.1 ValueArray28() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28>
testing::internal::ValueArray28< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28 >::ValueArray28 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
```

```
T6 v6,
T7 v7,
T8 v8,
T9 v9,
T10 v10,
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28) [inline]
```

8.170.1.2 ValueArray28() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28>
testing::internal::ValueArray28< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28 >::ValueArray28 (
    const ValueArray28< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28 > & other) [inline]
```

8.170.2 Dokumentacja funkcji składowych

8.170.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28>
template<typename T>
testing::internal::ValueArray28< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28 >::operator ParamGenerator<
T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.171 Dokumentacja szablonu klasy testing::internal::ValueArray29< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray29` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29)
- template<typename T>
 `operator ParamGenerator< T >` () const
- `ValueArray29` (const ValueArray29 &other)

8.171.1 Dokumentacja konstruktora i destruktora

8.171.1.1 ValueArray29() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29>
testing::internal::ValueArray29< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29 >::ValueArray29 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27,
    T28 v28,
    T29 v29) [inline]
```

8.171.1.2 ValueArray29() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29>
testing::internal::ValueArray29< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29 >::ValueArray29 (
    const ValueArray29< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29 > & other) [inline]
```

8.171.2 Dokumentacja funkcji składowych

8.171.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29>
template<typename T>
testing::internal::ValueArray29< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29 >::operator ParamGenerator<
T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.172 Dokumentacja szablonu klasy testing::internal::ValueArray3< T1, T2, T3 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- [ValueArray3](#) (T1 v1, T2 v2, T3 v3)
- template<typename T>
[operator ParamGenerator< T >](#) () const
- [ValueArray3](#) (const ValueArray3 &other)

8.172.1 Dokumentacja konstruktora i destruktora

8.172.1.1 ValueArray3() [1/2]

```
template<typename T1, typename T2, typename T3>
testing::internal::ValueArray3< T1, T2, T3 >::ValueArray3 (
    T1 v1,
    T2 v2,
    T3 v3) [inline]
```

8.172.1.2 ValueArray3() [2/2]

```
template<typename T1, typename T2, typename T3>
testing::internal::ValueArray3< T1, T2, T3 >::ValueArray3 (
    const ValueArray3< T1, T2, T3 > & other) [inline]
```

8.172.2 Dokumentacja funkcji składowych

8.172.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3>
template<typename T>
testing::internal::ValueArray3< T1, T2, T3 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.173 Dokumentacja szablonu klasy testing::internal::ValueArray30< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- [ValueArray30](#) (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30)
- template<typename T>
 [operator ParamGenerator< T >](#) () const
- [ValueArray30](#) (const ValueArray30 &other)

8.173.1 Dokumentacja konstruktora i destruktora

8.173.1.1 ValueArray30() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30>
testing::internal::ValueArray30< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30 >::ValueArray30 (
    T1 v1,
    T2 v2,
```

```
T3 v3,
T4 v4,
T5 v5,
T6 v6,
T7 v7,
T8 v8,
T9 v9,
T10 v10,
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30) [inline]
```

8.173.1.2 ValueArray30() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30>
testing::internal::ValueArray30< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30 >::ValueArray30
(
    const ValueArray30< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30 > & other)
[inline]
```

8.173.2 Dokumentacja funkcji składowych

8.173.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30>
template<typename T>
testing::internal::ValueArray30< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
```

```
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30 >::operator
ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.174 Dokumentacja szablonu klasy testing::internal::ValueArray31< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray31` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31)
- template<typename T>
`operator ParamGenerator< T > () const`
- `ValueArray31` (const ValueArray31 &other)

8.174.1 Dokumentacja konstruktora i destruktora

8.174.1.1 ValueArray31() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31>
testing::internal::ValueArray31< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31 >::ValueArray31 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
```

8.174 Dokumentacja szablonu klasy testing::internal::ValueArray31< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31 > 267

```
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30,
T31 v31) [inline]
```

8.174.1.2 ValueArray31() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31>
testing::internal::ValueArray31< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31 >::Value<-
Array31 (
    const ValueArray31< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31 > & other)
[inline]
```

8.174.2 Dokumentacja funkcji składowych

8.174.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31>
template<typename T>
testing::internal::ValueArray31< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31 >::operator
ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.175 Dokumentacja szablonu klasy testing::internal::ValueArray32< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray32` (`T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32)`
- template<typename T>
 `operator ParamGenerator< T > () const`
- `ValueArray32` (`const ValueArray32 &other`)

8.175.1 Dokumentacja konstruktora i destruktora

8.175.1.1 ValueArray32() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32>
testing::internal::ValueArray32< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32 >::←
ValueArray32 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
```

```
T24 v24,  
T25 v25,  
T26 v26,  
T27 v27,  
T28 v28,  
T29 v29,  
T30 v30,  
T31 v31,  
T32 v32) [inline]
```

8.175.1.2 ValueArray32() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32>  
testing::internal::ValueArray32< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32 >::←  
ValueArray32 (<  
    const ValueArray32< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32 > &  
other) [inline]
```

8.175.2 Dokumentacja funkcji składowych

8.175.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32>  
template<typename T>  
testing::internal::ValueArray32< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32 >::←  
::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.176 Dokumentacja szablonu klasy testing::internal::ValueArray33< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray33` (`T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33`)
- template<typename T>
 `operator ParamGenerator< T > () const`
- `ValueArray33` (`const ValueArray33 &other`)

8.176.1 Dokumentacja konstruktora i destruktora

8.176.1.1 `ValueArray33()` [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33>
testing::internal::ValueArray33< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33 >::ValueArray33 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27,
    T28 v28,
    T29 v29,
    T30 v30,
    T31 v31,
    T32 v32,
    T33 v33) [inline]
```

8.176.1.2 `ValueArray33()` [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33>
testing::internal::ValueArray33< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33
>::ValueArray33 (
    const ValueArray33< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33
> & other) [inline]
```

8.176.2 Dokumentacja funkcji składowych

8.176.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33>
template<typename T>
testing::internal::ValueArray33< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33
>::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.177 Dokumentacja szablonu klasy testing::internal::ValueArray34< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray34` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34)
- template<typename T>
 `operator ParamGenerator< T > () const`
- `ValueArray34` (const ValueArray34 &other)

8.177.1 Dokumentacja konstruktora i destruktora

8.177.1.1 ValueArray34() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34>  
testing::internal::ValueArray34< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,  
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,  
T34 >::ValueArray34 (   
    T1 v1,  
    T2 v2,  
    T3 v3,  
    T4 v4,  
    T5 v5,  
    T6 v6,  
    T7 v7,  
    T8 v8,  
    T9 v9,  
    T10 v10,  
    T11 v11,  
    T12 v12,  
    T13 v13,  
    T14 v14,  
    T15 v15,  
    T16 v16,  
    T17 v17,  
    T18 v18,  
    T19 v19,  
    T20 v20,  
    T21 v21,  
    T22 v22,  
    T23 v23,  
    T24 v24,  
    T25 v25,  
    T26 v26,  
    T27 v27,  
    T28 v28,  
    T29 v29,  
    T30 v30,  
    T31 v31,  
    T32 v32,  
    T33 v33,  
    T34 v34) [inline]
```

8.177.1.2 ValueArray34() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34>  
testing::internal::ValueArray34< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
```

8.178 Dokumentacja szablonu klasy testing::internal::ValueArray35< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35 >

273

```
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,  
T34 >::ValueArray34 <  
    const ValueArray34< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,  
    T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,  
    T34 > & other> [inline]
```

8.177.2 Dokumentacja funkcji składowych

8.177.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34>  
template<typename T>  
testing::internal::ValueArray34< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,  
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,  
T34 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.178 Dokumentacja szablonu klasy testing::internal::ValueArray35< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- **ValueArray35** (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35)
- template<typename T>
operator ParamGenerator< T > () const
- **ValueArray35** (const ValueArray35 &other)

8.178.1 Dokumentacja konstruktora i destruktora

8.178.1.1 ValueArray35() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35>  
testing::internal::ValueArray35< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,  
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,  
T34, T35 >::ValueArray35 (
```

T1 v1,
T2 v2,
T3 v3,
T4 v4,
T5 v5,
T6 v6,
T7 v7,
T8 v8,
T9 v9,
T10 v10,
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30,
T31 v31,
T32 v32,
T33 v33,
T34 v34,
T35 v35) [inline]

8.178.1.2 ValueArray35() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35>
```

8.179 Dokumentacja szablonu klasy testing::internal::ValueArray36< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36 >

275

```
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35>
testing::internal::ValueArray35< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35 >::ValueArray35 (
    const ValueArray35< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
    T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
    T34, T35 > & other) [inline]
```

8.178.2 Dokumentacja funkcji składowych

8.178.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35>
template<typename T>
testing::internal::ValueArray35< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.179 Dokumentacja szablonu klasy testing::internal::ValueArray36< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- **ValueArray36** (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36)
- template<typename T>
 operator ParamGenerator< T > () const
- **ValueArray36** (const ValueArray36 &other)

8.179.1 Dokumentacja konstruktora i destruktora

8.179.1.1 ValueArray36() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35, typename T36>  
testing::internal::ValueArray36< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,  
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,  
T34, T35, T36 >::ValueArray36 (  
    T1 v1,  
    T2 v2,  
    T3 v3,  
    T4 v4,  
    T5 v5,  
    T6 v6,  
    T7 v7,  
    T8 v8,  
    T9 v9,  
    T10 v10,  
    T11 v11,  
    T12 v12,  
    T13 v13,  
    T14 v14,  
    T15 v15,  
    T16 v16,  
    T17 v17,  
    T18 v18,  
    T19 v19,  
    T20 v20,  
    T21 v21,  
    T22 v22,  
    T23 v23,  
    T24 v24,  
    T25 v25,  
    T26 v26,  
    T27 v27,  
    T28 v28,  
    T29 v29,  
    T30 v30,  
    T31 v31,  
    T32 v32,  
    T33 v33,  
    T34 v34,  
    T35 v35,  
    T36 v36)  [inline]
```

8.179.1.2 ValueArray36() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
```

8.180 Dokumentacja szablonu klasy testing::internal::ValueArray37< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37 >

277

```
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36>
testing::internal::ValueArray36< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36 >::ValueArray36 (
    const ValueArray36< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36 > & other) [inline]
```

8.179.2 Dokumentacja funkcji składowych

8.179.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36>
template<typename T>
testing::internal::ValueArray36< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.180 Dokumentacja szablonu klasy testing::internal::ValueArray37< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- [ValueArray37](#) (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37)
- template<typename T>
[operator ParamGenerator< T > \(\) const](#)
- [ValueArray37](#) (const ValueArray37 &other)

8.180.1 Dokumentacja konstruktora i destruktora

8.180.1.1 ValueArray37() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35, typename T36, typename T37>  
testing::internal::ValueArray37< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,  
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,  
T34, T35, T36, T37 >::ValueArray37 (   
    T1 v1,  
    T2 v2,  
    T3 v3,  
    T4 v4,  
    T5 v5,  
    T6 v6,  
    T7 v7,  
    T8 v8,  
    T9 v9,  
    T10 v10,  
    T11 v11,  
    T12 v12,  
    T13 v13,  
    T14 v14,  
    T15 v15,  
    T16 v16,  
    T17 v17,  
    T18 v18,  
    T19 v19,  
    T20 v20,  
    T21 v21,  
    T22 v22,  
    T23 v23,  
    T24 v24,  
    T25 v25,  
    T26 v26,  
    T27 v27,  
    T28 v28,  
    T29 v29,  
    T30 v30,  
    T31 v31,  
    T32 v32,  
    T33 v33,  
    T34 v34,  
    T35 v35,  
    T36 v36,  
    T37 v37) [inline]
```

8.180.1.2 ValueArray37() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
```

8.181 Dokumentacja szablonu klasy testing::internal::ValueArray38< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38 >

279

```
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37>
testing::internal::ValueArray37< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37 >::ValueArray37 (
    const ValueArray37< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
    T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
    T34, T35, T36, T37 > & other) [inline]
```

8.180.2 Dokumentacja funkcji składowych

8.180.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37>
template<typename T>
testing::internal::ValueArray37< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.181 Dokumentacja szablonu klasy testing::internal::ValueArray38< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- [ValueArray38](#) (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38)
- template<typename T>
[operator ParamGenerator< T > \(\) const](#)
- [ValueArray38](#) (const ValueArray38 &other)

8.181.1 Dokumentacja konstruktora i destruktora

8.181.1.1 ValueArray38() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35, typename T36, typename T37, typename T38> ::ValueArray38(  
    T1 v1,  
    T2 v2,  
    T3 v3,  
    T4 v4,  
    T5 v5,  
    T6 v6,  
    T7 v7,  
    T8 v8,  
    T9 v9,  
    T10 v10,  
    T11 v11,  
    T12 v12,  
    T13 v13,  
    T14 v14,  
    T15 v15,  
    T16 v16,  
    T17 v17,  
    T18 v18,  
    T19 v19,  
    T20 v20,  
    T21 v21,  
    T22 v22,  
    T23 v23,  
    T24 v24,  
    T25 v25,  
    T26 v26,  
    T27 v27,  
    T28 v28,  
    T29 v29,  
    T30 v30,  
    T31 v31,  
    T32 v32,  
    T33 v33,  
    T34 v34,  
    T35 v35,  
    T36 v36,  
    T37 v37,  
    T38 v38) [inline]
```

8.181.1.2 ValueArray38() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
```

8.182 Dokumentacja szablonu klasy testing::internal::ValueArray39< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39 >

281

```
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38>
testing::internal::ValueArray38< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38 >::ValueArray38 (
    const ValueArray38< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38 > & other) [inline]
```

8.181.2 Dokumentacja funkcji składowych

8.181.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38>
template<typename T>
testing::internal::ValueArray38< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.182 Dokumentacja szablonu klasy testing::internal::ValueArray39< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- [ValueArray39](#) (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39)
- template<typename T>
[operator ParamGenerator< T > \(\) const](#)
- [ValueArray39](#) (const ValueArray39 &other)

8.182.1 Dokumentacja konstruktora i destruktora

8.182.1.1 ValueArray39() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35, typename T36, typename T37, typename T38, typename T39>  
testing::internal::ValueArray39< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,  
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,  
T34, T35, T36, T37, T38, T39 >::ValueArray39 (
```

T1 v1,
T2 v2,
T3 v3,
T4 v4,
T5 v5,
T6 v6,
T7 v7,
T8 v8,
T9 v9,
T10 v10,
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30,
T31 v31,
T32 v32,
T33 v33,
T34 v34,
T35 v35,
T36 v36,
T37 v37,
T38 v38,
T39 v39) [inline]

8.182.1.2 ValueArray39() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
```

```
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39>
testing::internal::ValueArray39< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39 >::ValueArray39>
    const ValueArray39< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39 > & other) [inline]
```

8.182.2 Dokumentacja funkcji składowych

8.182.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39>
template<typename T>
testing::internal::ValueArray39< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.183 Dokumentacja szablonu klasy testing::internal::ValueArray4< T1, T2, T3, T4 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- **ValueArray4** (T1 v1, T2 v2, T3 v3, T4 v4)
- template<typename T>
 operator ParamGenerator< T > () const
- **ValueArray4** (const ValueArray4 &other)

8.183.1 Dokumentacja konstruktora i destruktora

8.183.1.1 ValueArray4() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4>
testing::internal::ValueArray4< T1, T2, T3, T4 >::ValueArray4 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4) [inline]
```

8.183.1.2 ValueArray4() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4>
testing::internal::ValueArray4< T1, T2, T3, T4 >::ValueArray4 (
    const ValueArray4< T1, T2, T3, T4 > & other) [inline]
```

8.183.2 Dokumentacja funkcji składowych

8.183.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4>
template<typename T>
testing::internal::ValueArray4< T1, T2, T3, T4 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.184 Dokumentacja szablonu klasy testing::internal::ValueArray40< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- [ValueArray40](#) (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40)
- template<typename T>
[operator ParamGenerator< T >](#) () const
- [ValueArray40](#) (const ValueArray40 &other)

8.184.1 Dokumentacja konstruktora i destruktora

8.184.1.1 ValueArray40() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40>
testing::internal::ValueArray40< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40 >::ValueArray40 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27,
    T28 v28,
    T29 v29,
    T30 v30,
    T31 v31,
    T32 v32,
    T33 v33,
    T34 v34,
    T35 v35,
    T36 v36,
    T37 v37,
    T38 v38,
    T39 v39,
    T40 v40) [inline]
```

8.184.1.2 ValueArray40() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14>
```

```
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40>
testing::internal::ValueArray40< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40 >::ValueArray40 (
    const ValueArray40< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40 > & other) [inline]
```

8.184.2 Dokumentacja funkcji składowych

8.184.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40>
template<typename T>
testing::internal::ValueArray40< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/[gtest-param-util.h](#)

8.185 Dokumentacja szablonu klasy **testing::internal::ValueArray41< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41 >**

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- [ValueArray41](#) (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41)
- template<typename T>
[operator ParamGenerator< T > \(\) const](#)
- [ValueArray41](#) (const ValueArray41 &other)

8.185.1.1 ValueArray41() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41>
testing::internal::ValueArray41< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41 >::ValueArray41 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27,
    T28 v28,
    T29 v29,
    T30 v30,
    T31 v31,
    T32 v32,
    T33 v33,
    T34 v34,
    T35 v35,
    T36 v36,
    T37 v37,
    T38 v38,
    T39 v39,
    T40 v40,
    T41 v41) [inline]
```

8.185.1.2 ValueArray41() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41 >::ValueArray41
```

```

T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41>
testing::internal::ValueArray41< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41 >::ValueArray41 (
    const ValueArray41< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41 > & other) [inline]
```

8.185.2 Dokumentacja funkcji składowych

8.185.2.1 operator ParamGenerator< T >()

```

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41>
template<typename T>
testing::internal::ValueArray41< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.186 Dokumentacja szablonu klasy **testing::internal::ValueArray42< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42 >**

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- **ValueArray42** (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42)
- template<typename T>
operator ParamGenerator< T > () const
- **ValueArray42** (const ValueArray42 &other)

8.186.1 Dokumentacja konstruktora i destruktora

8.186.1.1 ValueArray42() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42>
testing::internal::ValueArray42< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42 >::ValueArray42 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27,
    T28 v28,
    T29 v29,
    T30 v30,
    T31 v31,
    T32 v32,
    T33 v33,
    T34 v34,
    T35 v35,
    T36 v36,
    T37 v37,
    T38 v38,
    T39 v39,
    T40 v40,
    T41 v41,
    T42 v42) [inline]
```

8.186.1.2 `ValueArray42()` [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename
T42>
testing::internal::ValueArray42< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42 >::ValueArray42 (
    const ValueArray42< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42 > & other) [inline]
```

8.186.2 Dokumentacja funkcji składowych

8.186.2.1 `operator ParamGenerator< T >()`

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename
T42>
template<typename T>
testing::internal::ValueArray42< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32,
T33, T34, T35, T36, T37, T38, T39, T40, T41, T42 >::operator ParamGenerator< T > () const
[inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.187 Dokumentacja szablonu klasy `testing::internal::ValueArray43< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43 >`

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray43` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43)
- template<typename T>
`operator ParamGenerator< T > () const`
- `ValueArray43` (const ValueArray43 &other)

8.187.1 Dokumentacja konstruktora i destruktora

8.187.1.1 ValueArray43() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43>
testing::internal::ValueArray43< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43 >::ValueArray43 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27,
    T28 v28,
    T29 v29,
    T30 v30,
    T31 v31,
    T32 v32,
    T33 v33,
    T34 v34,
    T35 v35,
    T36 v36,
    T37 v37,
    T38 v38,
    T39 v39,
    T40 v40,
    T41 v41,
    T42 v42,
    T43 v43) [inline]
```

8.187.1.2 `ValueArray43()` [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename
T42, typename T43>
testing::internal::ValueArray43< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43 >::ValueArray43 (
    const ValueArray43< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43 > & other) [inline]
```

8.187.2 Dokumentacja funkcji składowych

8.187.2.1 `operator ParamGenerator< T >()`

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename
T42, typename T43>
template<typename T>
testing::internal::ValueArray43< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43 >::operator ParamGenerator< T > () const
[inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.188 Dokumentacja szablonu klasy `testing::internal::ValueArray44< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44 >`

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray44` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44)
- template<typename T>
`operator ParamGenerator< T > () const`
- `ValueArray44` (const `ValueArray44 &other`)

8.188.1 Dokumentacja konstruktora i destruktora

8.188.1.1 ValueArray44() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44>
testing::internal::ValueArray44< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44 >::ValueArray44 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27,
    T28 v28,
    T29 v29,
    T30 v30,
    T31 v31,
    T32 v32,
    T33 v33,
    T34 v34,
    T35 v35,
    T36 v36,
    T37 v37,
    T38 v38,
    T39 v39,
    T40 v40,
    T41 v41,
    T42 v42,
    T43 v43,
    T44 v44) [inline]
```

8.188.1.2 ValueArray44() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44>
testing::internal::ValueArray44< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44 >::ValueArray44 (
    const ValueArray44< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44 > & other) [inline]
```

8.188.2 Dokumentacja funkcji składowych

8.188.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44>
template<typename T>
testing::internal::ValueArray44< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44 >::operator ParamGenerator< T > ()
const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.189 Dokumentacja szablonu klasy testing::internal::ValueArray45< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45 >

```
#include <gtest-param-util-generated.h>
```

- [ValueArray45](#) (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45)
- template<typename T>
[operator ParamGenerator< T >](#) () const
- [ValueArray45](#) (const ValueArray45 &other)

8.189.1 Dokumentacja konstruktora i destruktora

8.189.1.1 ValueArray45() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44, typename T45>
testing::internal::ValueArray45< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45 >::ValueArray45 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27,
    T28 v28,
    T29 v29,
    T30 v30,
    T31 v31,
```

```

T32 v32,
T33 v33,
T34 v34,
T35 v35,
T36 v36,
T37 v37,
T38 v38,
T39 v39,
T40 v40,
T41 v41,
T42 v42,
T43 v43,
T44 v44,
T45 v45) [inline]

```

8.189.1.2 **ValueArray45()** [2/2]

```

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename
T42, typename T43, typename T44, typename T45>
testing::internal::ValueArray45< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45 >::ValueArray45 (
    const ValueArray45< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45 > & other) [inline]

```

8.189.2 Dokumentacja funkcji składowych

8.189.2.1 **operator ParamGenerator< T >()**

```

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename
T42, typename T43, typename T44, typename T45>
template<typename T>
testing::internal::ValueArray45< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45 >::operator ParamGenerator< T > ()
const [inline]

```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.190 Dokumentacja szablonu klasy testing::internal::ValueArray46< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46 >
8.190 Dokumentacja szablonu klasy testing::internal::ValueArray46< T1, 297

T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray46` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46)
- template<typename T>
`operator ParamGenerator< T > () const`
- `ValueArray46` (const ValueArray46 &other)

8.190.1 Dokumentacja konstruktora i destruktora

8.190.1.1 ValueArray46() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44, typename T45, typename T46>
testing::internal::ValueArray46< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46 >::ValueArray46 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
```

```

T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30,
T31 v31,
T32 v32,
T33 v33,
T34 v34,
T35 v35,
T36 v36,
T37 v37,
T38 v38,
T39 v39,
T40 v40,
T41 v41,
T42 v42,
T43 v43,
T44 v44,
T45 v45,
T46 v46) [inline]

```

8.190.1.2 ValueArray46() [2/2]

```

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename
T42, typename T43, typename T44, typename T45, typename T46>
testing::internal::ValueArray46< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46 >::ValueArray46 (
    const ValueArray46< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46 > & other) [inline]

```

8.190.2 Dokumentacja funkcji składowych

8.190.2.1 operator ParamGenerator< T >()

```

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename
T42, typename T43, typename T44, typename T45, typename T46>
operator ParamGenerator< T >()

```

```
T42, typename T43, typename T44, typename T45, typename T46>
template<typename T>
testing::internal::ValueArray46< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46 >::operator ParamGenerator< T
> () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.191 Dokumentacja szablonu klasy testing::internal::ValueArray47< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- **ValueArray47** (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47)
- template<typename T>
operator ParamGenerator< T > () const
- **ValueArray47** (const ValueArray47 &other)

8.191.1 Dokumentacja konstruktora i destruktora

8.191.1.1 ValueArray47() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename
T42, typename T43, typename T44, typename T45, typename T46, typename T47>
testing::internal::ValueArray47< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47 >::ValueArray47 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
```

```

T7 v7,
T8 v8,
T9 v9,
T10 v10,
T11 v11,
T12 v12,
T13 v13,
T14 v14,
T15 v15,
T16 v16,
T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30,
T31 v31,
T32 v32,
T33 v33,
T34 v34,
T35 v35,
T36 v36,
T37 v37,
T38 v38,
T39 v39,
T40 v40,
T41 v41,
T42 v42,
T43 v43,
T44 v44,
T45 v45,
T46 v46,
T47 v47) [inline]

```

8.191.1.2 ValueArray47() [2/2]

```

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename
T42, typename T43, typename T44, typename T45, typename T46, typename T47>
testing::internal::ValueArray47< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47 >::ValueArray47 (
    const ValueArray47< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47 > & other) [inline]

```

8.191.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44, typename T45, typename T46, typename T47>
template<typename T>
testing::internal::ValueArray47< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.192 Dokumentacja szablonu klasy testing::internal::ValueArray48< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- [ValueArray48 \(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47, T48 v48\)](#)
- template<typename T>
 [operator ParamGenerator< T > \(\) const](#)
- [ValueArray48 \(const ValueArray48 &other\)](#)

8.192.1 Dokumentacja konstruktora i destruktora

8.192.1.1 ValueArray48() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename  
T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48>  
testing::internal::ValueArray48< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,  
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,  
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48 >::ValueArray48 (   
T1 v1,  
T2 v2,  
T3 v3,  
T4 v4,  
T5 v5,  
T6 v6,  
T7 v7,  
T8 v8,  
T9 v9,  
T10 v10,  
T11 v11,  
T12 v12,  
T13 v13,  
T14 v14,  
T15 v15,  
T16 v16,  
T17 v17,  
T18 v18,  
T19 v19,  
T20 v20,  
T21 v21,  
T22 v22,  
T23 v23,  
T24 v24,  
T25 v25,  
T26 v26,  
T27 v27,  
T28 v28,  
T29 v29,  
T30 v30,  
T31 v31,  
T32 v32,  
T33 v33,  
T34 v34,  
T35 v35,  
T36 v36,  
T37 v37,  
T38 v38,  
T39 v39,  
T40 v40,  
T41 v41,  
T42 v42,  
T43 v43,
```

```
T44 v44,  
T45 v45,  
T46 v46,  
T47 v47,  
T48 v48) [inline]
```

8.192.1.2 ValueArray48() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48>  
testing::internal::ValueArray48< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48 >::ValueArray48 (   
    const ValueArray48< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48 > & other) [inline]
```

8.192.2 Dokumentacja funkcji składowych

8.192.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48>  
template<typename T>  
testing::internal::ValueArray48< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48 >::operator  
ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.193 Dokumentacja szablonu klasy testing::internal::ValueArray49< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray49` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47, T48 v48, T49 v49)
- template<typename T>
operator `ParamGenerator< T >` () const
- `ValueArray49` (const `ValueArray49` &other)

8.193.1 Dokumentacja konstruktora i destruktora

8.193.1.1 `ValueArray49()` [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48, typename T49>
testing::internal::ValueArray49< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49 >::ValueArray49
(
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
    T17 v17,
    T18 v18,
    T19 v19,
    T20 v20,
    T21 v21,
    T22 v22,
    T23 v23,
    T24 v24,
    T25 v25,
    T26 v26,
    T27 v27,
    T28 v28,
```

```
T29 v29,  
T30 v30,  
T31 v31,  
T32 v32,  
T33 v33,  
T34 v34,  
T35 v35,  
T36 v36,  
T37 v37,  
T38 v38,  
T39 v39,  
T40 v40,  
T41 v41,  
T42 v42,  
T43 v43,  
T44 v44,  
T45 v45,  
T46 v46,  
T47 v47,  
T48 v48,  
T49 v49) [inline]
```

8.193.1.2 ValueArray49() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename  
T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48, typename  
T49>  
testing::internal::ValueArray49< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,  
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,  
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49 >::ValueArray49  
(  
    const ValueArray49< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,  
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,  
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49 > & other)  
[inline]
```

8.193.2 Dokumentacja funkcji składowych

8.193.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename  
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename  
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename  
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename  
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename  
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename  
T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48, typename  
T49>  
template<typename T>
```

```
testing::internal::ValueArray49< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49 >::operator
ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.194 Dokumentacja szablonu klasy testing::internal::ValueArray5< T1, T2, T3, T4, T5 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray5` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5)
- template<typename T>
 `operator ParamGenerator< T >` () const
- `ValueArray5` (const `ValueArray5` &other)

8.194.1 Dokumentacja konstruktora i destruktora

8.194.1.1 `ValueArray5()` [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5>
testing::internal::ValueArray5< T1, T2, T3, T4, T5 >::ValueArray5 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5) [inline]
```

8.194.1.2 `ValueArray5()` [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5>
testing::internal::ValueArray5< T1, T2, T3, T4, T5 >::ValueArray5 (
    const ValueArray5< T1, T2, T3, T4, T5 > & other) [inline]
```

8.194.2 Dokumentacja funkcji składowych

8.194.2.1 `operator ParamGenerator< T >()`

```
template<typename T1, typename T2, typename T3, typename T4, typename T5>
template<typename T>
testing::internal::ValueArray5< T1, T2, T3, T4, T5 >::operator ParamGenerator< T > () const
[inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.195 Dokumentacja szablonu klasy testing::internal::ValueArray50< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49, T50 >

8.195 Dokumentacja szablonu klasy testing::internal::ValueArray50< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49, T50 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray50` (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47, T48 v48, T49 v49, T50 v50)
- template<typename T>
 `operator ParamGenerator< T >` () const
- `ValueArray50` (const ValueArray50 &other)

8.195.1 Dokumentacja konstruktora i destruktora

8.195.1.1 ValueArray50() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48, typename T49, typename T50>
testing::internal::ValueArray50< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49, T50 >::ValueArray50 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9,
    T10 v10,
    T11 v11,
    T12 v12,
    T13 v13,
    T14 v14,
    T15 v15,
    T16 v16,
```

```

T17 v17,
T18 v18,
T19 v19,
T20 v20,
T21 v21,
T22 v22,
T23 v23,
T24 v24,
T25 v25,
T26 v26,
T27 v27,
T28 v28,
T29 v29,
T30 v30,
T31 v31,
T32 v32,
T33 v33,
T34 v34,
T35 v35,
T36 v36,
T37 v37,
T38 v38,
T39 v39,
T40 v40,
T41 v41,
T42 v42,
T43 v43,
T44 v44,
T45 v45,
T46 v46,
T47 v47,
T48 v48,
T49 v49,
T50 v50) [inline]

```

8.195.1.2 ValueArray50() [2/2]

```

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename
T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename
T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename
T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename
T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48, typename
T49, typename T50>
testing::internal::ValueArray50< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49, T50 >::Value<-
Array50 (
    const ValueArray50< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33,
T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49, T50 > & other)
[inline]

```

8.195.2 Dokumentacja funkcji składowych

8.195.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48, typename T49, typename T50>
template<typename T>
testing::internal::ValueArray50< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49, T50 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.196 Dokumentacja szablonu klasy testing::internal::ValueArray6< T1, T2, T3, T4, T5, T6 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- [ValueArray6](#) (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6)
- template<typename T>
[operator ParamGenerator< T >](#) () const
- [ValueArray6](#) (const ValueArray6 &other)

8.196.1 Dokumentacja konstruktora i destruktora

8.196.1.1 ValueArray6() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>
testing::internal::ValueArray6< T1, T2, T3, T4, T5, T6 >::ValueArray6 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6) [inline]
```

8.196.1.2 `ValueArray6()` [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>
testing::internal::ValueArray6< T1, T2, T3, T4, T5, T6 >::ValueArray6 (
    const ValueArray6< T1, T2, T3, T4, T5, T6 > & other) [inline]
```

8.196.2 Dokumentacja funkcji składowych

8.196.2.1 `operator ParamGenerator< T >()`

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>
template<typename T>
testing::internal::ValueArray6< T1, T2, T3, T4, T5, T6 >::operator ParamGenerator< T > ()
const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-generated.h

8.197 Dokumentacja szablonu klasy `testing::internal::ValueArray7< T1, T2, T3, T4, T5, T6, T7 >`

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray7` (`T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7`)
- `template<typename T> operator ParamGenerator< T > () const`
- `ValueArray7 (const ValueArray7 &other)`

8.197.1 Dokumentacja konstruktora i destruktora

8.197.1.1 `ValueArray7()` [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7>
testing::internal::ValueArray7< T1, T2, T3, T4, T5, T6, T7 >::ValueArray7 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7) [inline]
```

8.197.1.2 ValueArray7() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7>
testing::internal::ValueArray7< T1, T2, T3, T4, T5, T6, T7 >::ValueArray7 (
    const ValueArray7< T1, T2, T3, T4, T5, T6, T7 > & other) [inline]
```

8.197.2 Dokumentacja funkcji składowych

8.197.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7>
template<typename T>
testing::internal::ValueArray7< T1, T2, T3, T4, T5, T6, T7 >::operator ParamGenerator< T > () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-generated.h

8.198 Dokumentacja szablonu klasy testing::internal::ValueArray8< T1, T2, T3, T4, T5, T6, T7, T8 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- **ValueArray8** (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8)
- template<typename T>
 operator ParamGenerator< T > () const
- **ValueArray8** (const ValueArray8 &other)

8.198.1 Dokumentacja konstruktora i destruktora

8.198.1.1 ValueArray8() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8>
testing::internal::ValueArray8< T1, T2, T3, T4, T5, T6, T7, T8 >::ValueArray8 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8) [inline]
```

8.198.1.2 ValueArray8() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8>
testing::internal::ValueArray8< T1, T2, T3, T4, T5, T6, T7, T8 >::ValueArray8 (
    const ValueArray8< T1, T2, T3, T4, T5, T6, T7, T8 > & other) [inline]
```

8.198.2 Dokumentacja funkcji składowych

8.198.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8>
template<typename T>
testing::internal::ValueArray8< T1, T2, T3, T4, T5, T6, T7, T8 >::operator ParamGenerator< T
> () const [inline]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-generated.h

8.199 Dokumentacja szablonu klasy testing::internal::ValueArray9< T1, T2, T3, T4, T5, T6, T7, T8, T9 >

```
#include <gtest-param-util-generated.h>
```

Metody publiczne

- `ValueArray9` (`T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9`)
- `template<typename T>`
`operator ParamGenerator< T > () const`
- `ValueArray9` (`const ValueArray9 &other`)

8.199.1 Dokumentacja konstruktora i destruktora

8.199.1.1 ValueArray9() [1/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename
T7, typename T8, typename T9>
testing::internal::ValueArray9< T1, T2, T3, T4, T5, T6, T7, T8, T9 >::ValueArray9 (
    T1 v1,
    T2 v2,
    T3 v3,
    T4 v4,
    T5 v5,
    T6 v6,
    T7 v7,
    T8 v8,
    T9 v9) [inline]
```

8.199.1.2 ValueArray9() [2/2]

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9>
testing::internal::ValueArray9< T1, T2, T3, T4, T5, T6, T7, T8, T9 >::ValueArray9 (
    const ValueArray9< T1, T2, T3, T4, T5, T6, T7, T8, T9 > & other) [inline]
```

8.199.2 Dokumentacja funkcji składowych

8.199.2.1 operator ParamGenerator< T >()

```
template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9>
template<typename T>
testing::internal::ValueArray9< T1, T2, T3, T4, T5, T6, T7, T8, T9 >::operator ParamGenerator< T > () const [inline]
```

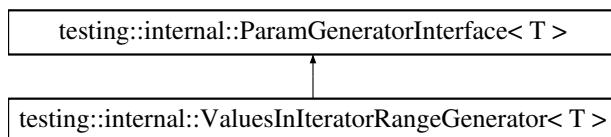
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.200 Dokumentacja szablonu klasy testing::internal::ValuesInIteratorRangeGenerator< T >

```
#include <gtest-param-util.h>
```

Diagram dziedziczenia dla testing::internal::ValuesInIteratorRangeGenerator< T >



Metody publiczne

- template<typename ForwardIterator>
[ValuesInIteratorRangeGenerator](#) (ForwardIterator begin, ForwardIterator end)
- virtual ~[ValuesInIteratorRangeGenerator](#) ()
- virtual [ParamIteratorInterface< T > * Begin](#) () const
- virtual [ParamIteratorInterface< T > * End](#) () const

Metody publiczne dziedziczone z testing::internal::ParamGeneratorInterface< T >

- virtual ~[ParamGeneratorInterface](#) ()

Dodatkowe dziedziczone składowe

Typy publiczne dziedziczone z `testing::internal::ParamGeneratorInterface< T >`

- `typedef T ParamType`

8.200.1 Dokumentacja konstruktora i destruktora

8.200.1.1 `ValuesInIteratorRangeGenerator()`

```
template<typename T>
template<typename ForwardIterator>
testing::internal::ValuesInIteratorRangeGenerator< T >::ValuesInIteratorRangeGenerator (
    ForwardIterator begin,
    ForwardIterator end) [inline]
```

8.200.1.2 `~ValuesInIteratorRangeGenerator()`

```
template<typename T>
virtual testing::internal::ValuesInIteratorRangeGenerator< T >::~ValuesInIteratorRangeGenerator
() [inline], [virtual]
```

8.200.2 Dokumentacja funkcji składowych

8.200.2.1 `Begin()`

```
template<typename T>
virtual ParamIteratorInterface< T > * testing::internal::ValuesInIteratorRangeGenerator< T
>::Begin () const [inline], [virtual]
```

Implementuje `testing::internal::ParamGeneratorInterface< T >`.

8.200.2.2 `End()`

```
template<typename T>
virtual ParamIteratorInterface< T > * testing::internal::ValuesInIteratorRangeGenerator< T
>::End () const [inline], [virtual]
```

Implementuje `testing::internal::ParamGeneratorInterface< T >`.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

8.201 Dokumentacja szablonu struktury testing::internal::VoidT< T >

```
#include <gtest-internal.h>
```

Typy publiczne

- `typedef void value_type`

8.201.1 Dokumentacja składowych definicji typu

8.201.1.1 value_type

```
template<typename T>
typedef void testing::internal::VoidT< T >::value_type
```

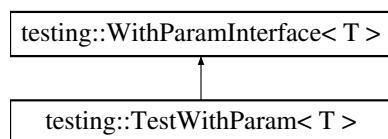
Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

8.202 Dokumentacja szablonu klasy testing::WithParamInterface< T >

```
#include <gtest.h>
```

Diagram dziedziczenia dla testing::WithParamInterface< T >



Typy publiczne

- `typedef T ParamType`

Metody publiczne

- `virtual ~WithParamInterface ()`
- `const ParamType & GetParam () const`

Przyjaciele

- `template<class TestClass>
class internal::ParameterizedTestFactory`

8.202.1 Dokumentacja składowych definicji typu

8.202.1.1 ParamType

```
template<typename T>
typedef T testing::WithParamInterface< T >::ParamType
```

8.202.2 Dokumentacja konstruktora i destruktora

8.202.2.1 ~WithParamInterface()

```
template<typename T>
virtual testing::WithParamInterface< T >::~WithParamInterface () [inline], [virtual]
```

8.202.3 Dokumentacja funkcji składowych

8.202.3.1 GetParam()

```
template<typename T>
const ParamType & testing::WithParamInterface< T >::GetParam () const [inline]
```

8.202.4 Dokumentacja przyjaciół i powiązanych symboli

8.202.4.1 internal::ParameterizedTestFactory

```
template<typename T>
template<class TestClass>
friend class internal::ParameterizedTestFactory [friend]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

8.203 Dokumentacja szablonu struktury

testing::internal::WrapPrinterType< type >

```
#include <gtest-printers.h>
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

Rozdział 9

Dokumentacja plików

9.1 Dokumentacja pliku MergeSortApp/MergeSort.h

Implementacja algorytmu sortowania przez scalanie (Merge Sort).

```
#include <vector>
```

Komponenty

- class `MergeSort< T >`
Klasa szablonowa realizująca algorytm sortowania przez scalanie.

9.1.1 Opis szczegółowy

Implementacja algorytmu sortowania przez scalanie (Merge Sort).

9.2 MergeSort.h

[Idź do dokumentacji tego pliku.](#)

```
00001
00005 #ifndef MERGESORT_H
00006 #define MERGESORT_H
00007
00008 #include <vector>
00009
00015 template<typename T>
00016 class MergeSort {
00017 public:
00018     static void sort(std::vector<T>& arr) {
00019         if (arr.size() <= 1) return;
00020         mergeSort(arr, 0, static_cast<int>(arr.size()) - 1);
00021     }
00022
00023 private:
00024     static void mergeSort(std::vector<T>& arr, int left, int right) {
00025         if (left >= right) return;
00026
00027         int mid = left + (right - left) / 2;
00028         mergeSort(arr, left, mid);
00029         mergeSort(arr, mid + 1, right);
```

```

00030     merge(arr, left, mid, right);
00031 }
00032
00033 static void merge(std::vector<T>& arr, int left, int mid, int right) {
00034     std::vector<T> temp(right - left + 1);
00035     int i = left, j = mid + 1, k = 0;
00036
00037     while (i <= mid && j <= right)
00038         temp[k++] = (arr[i] < arr[j]) ? arr[i++] : arr[j++];
00039
00040     while (i <= mid) temp[k++] = arr[i++];
00041     while (j <= right) temp[k++] = arr[j++];
00042
00043     for (int x = 0; x < k; x++)
00044         arr[left + x] = temp[x];
00045 }
00046 };
00047 #endif

```

9.3 Dokumentacja pliku MergeSortApp/MergeSortApp.cpp

Główny plik programu uruchamiający algorytm.

```
#include <iostream>
#include <vector>
#include "MergeSort.h"
```

Funkcje

- int `main ()`

9.3.1 Opis szczegółowy

Główny plik programu uruchamiający algorytm.

Autor

Marecik Michał

9.3.2 Dokumentacja funkcji

9.3.2.1 `main()`

```
int main ()
```

9.4 Dokumentacja pliku

packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-death-test.h

```
#include "gtest/internal/gtest-death-test-internal.h"
```

Przestrzenie nazw

- namespace `testing`

Definicje

- `#define GTEST_UNSUPPORTED_DEATH_TEST(statement, regex, terminator)`
- `#define EXPECT_DEATH_IF_SUPPORTED(statement, regex)`
- `#define ASSERT_DEATH_IF_SUPPORTED(statement, regex)`

Funkcje

- `testing::GTEST_DECLARE_string_(death_test_style)`

9.4.1 Dokumentacja definicji

9.4.1.1 ASSERT_DEATH_IF_SUPPORTED

```
#define ASSERT_DEATH_IF_SUPPORTED( \
    statement, \
    regex)
```

Wartość:

```
GTEST_UNSUPPORTED_DEATH_TEST(statement, regex, return)
```

9.4.1.2 EXPECT_DEATH_IF_SUPPORTED

```
#define EXPECT_DEATH_IF_SUPPORTED( \
    statement, \
    regex)
```

Wartość:

```
GTEST_UNSUPPORTED_DEATH_TEST(statement, regex, )
```

9.4.1.3 GTEST_UNSUPPORTED_DEATH_TEST

```
#define GTEST_UNSUPPORTED_DEATH_TEST( \
    statement, \
    regex, \
    terminator)
```

Wartość:

```
GTEST_AMBIGUOUS_ELSE_BLOCKER_ \
if (::testing::internal::AlwaysTrue()) { \
    GTEST_LOG_(WARNING) \
        << "Death tests are not supported on this platform.\n" \
        << "Statement '" #statement "' cannot be verified."; \
} else if (::testing::internal::AlwaysFalse()) { \
    ::testing::internal::RE::PartialMatch(".*", (regex)); \
    GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement); \
    terminator; \
} else \
    ::testing::Message()
```

9.5 gtest-death-test.h

[Idź do dokumentacji tego pliku.](#)

```

00001 // Copyright 2005, Google Inc.
00002 // All rights reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //      * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //      * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 //
00031 // The Google C++ Testing and Mocking Framework (Google Test)
00032 //
00033 // This header file defines the public API for death tests. It is
00034 // #included by gtest.h so a user doesn't need to include this
00035 // directly.
00036 // GOOGLETEST_CM0001 DO NOT DELETE
00037
00038 #ifndef GTEST_INCLUDE_GTEST_GTEST_DEATH_TEST_H_
00039 #define GTEST_INCLUDE_GTEST_GTEST_DEATH_TEST_H_
00040
00041 #include "gtest/internal/gtest-death-test-internal.h"
00042
00043 namespace testing {
00044
00045 // This flag controls the style of death tests. Valid values are "threadsafe",
00046 // meaning that the death test child process will re-execute the test binary
00047 // from the start, running only a single death test, or "fast",
00048 // meaning that the child process will execute the test logic immediately
00049 // after forking.
00050 GTEST_DECLARE_string_(death_test_style);
00051
00052 #if GTEST_HAS_DEATH_TEST
00053
00054 namespace internal {
00055
00056 // Returns a Boolean value indicating whether the caller is currently
00057 // executing in the context of the death test child process. Tools such as
00058 // Valgrind heap checkers may need this to modify their behavior in death
00059 // tests. IMPORTANT: This is an internal utility. Using it may break the
00060 // implementation of death tests. User code MUST NOT use it.
00061 GTEST_API_ bool InDeathTestChild();
00062
00063 } // namespace internal
00064
00065 // The following macros are useful for writing death tests.
00066
00067 // Here's what happens when an ASSERT_DEATH* or EXPECT_DEATH* is
00068 // executed:
00069 //
00070 // 1. It generates a warning if there is more than one active
00071 // thread. This is because it's safe to fork() or clone() only
00072 // when there is a single thread.
00073 //
00074 // 2. The parent process clone()s a sub-process and runs the death
00075 // test in it; the sub-process exits with code 0 at the end of the
00076 // death test, if it hasn't exited already.
00077 //
00078 // 3. The parent process waits for the sub-process to terminate.
00079 //
00080 // 4. The parent process checks the exit code and error message of
00081 // the sub-process.
00082 //

```

```
00083 // Examples:
00084 //
00085 // ASSERT_DEATH(server.SendMessage(56, "Hello"), "Invalid port number");
00086 // for (int i = 0; i < 5; i++) {
00087 //     EXPECT_DEATH(server.ProcessRequest(i),
00088 //         "Invalid request .* in ProcessRequest()")
00089 //         « "Failed to die on request " « i;
00090 // }
00091 //
00092 // ASSERT_EXIT(server.ExitNow(), ::testing::ExitedWithCode(0), "Exiting");
00093 //
00094 // bool KilledBySIGHUP(int exit_code) {
00095 //     return WIFSIGNALED(exit_code) && WTERMSIG(exit_code) == SIGHUP;
00096 // }
00097 //
00098 // ASSERT_EXIT(client.HangUpServer(), KilledBySIGHUP, "Hanging up!");
00099 //
00100 // On the regular expressions used in death tests:
00101 //
00102 // GOOGLETEST_CM0005 DO NOT DELETE
00103 // On POSIX-compliant systems (*nix), we use the <regex.h> library,
00104 // which uses the POSIX extended regex syntax.
00105 //
00106 // On other platforms (e.g. Windows or Mac), we only support a simple regex
00107 // syntax implemented as part of Google Test. This limited
00108 // implementation should be enough most of the time when writing
00109 // death tests; though it lacks many features you can find in PCRE
00110 // or POSIX extended regex syntax. For example, we don't support
00111 // union ("x|y"), grouping ("(xy)"), brackets ("[xy]"), and
00112 // repetition count ("x{5,7}"), among others.
00113 //
00114 // Below is the syntax that we do support. We chose it to be a
00115 // subset of both PCRE and POSIX extended regex, so it's easy to
00116 // learn wherever you come from. In the following: 'A' denotes a
00117 // literal character, period (.), or a single \\ escape sequence;
00118 // 'x' and 'y' denote regular expressions; 'm' and 'n' are for
00119 // natural numbers.
00120 //
00121 // c    matches any literal character c
00122 // \\\d  matches any decimal digit
00123 // \\\D  matches any character that's not a decimal digit
00124 // \\\f  matches \f
00125 // \\\n  matches \n
00126 // \\\r  matches \r
00127 // \\\s  matches any ASCII whitespace, including \n
00128 // \\\S  matches any character that's not a whitespace
00129 // \\\t  matches \t
00130 // \\\v  matches \v
00131 // \\\w  matches any letter, _, or decimal digit
00132 // \\\W  matches any character that \\\w doesn't match
00133 // \\\c  matches any literal character c, which must be a punctuation
00134 // .   matches any single character except \n
00135 // A?  matches 0 or 1 occurrences of A
00136 // A*  matches 0 or many occurrences of A
00137 // A+  matches 1 or many occurrences of A
00138 // ^   matches the beginning of a string (not that of each line)
00139 // $   matches the end of a string (not that of each line)
00140 // xy  matches x followed by y
00141 //
00142 // If you accidentally use PCRE or POSIX extended regex features
00143 // not implemented by us, you will get a run-time failure. In that
00144 // case, please try to rewrite your regular expression within the
00145 // above syntax.
00146 //
00147 // This implementation is *not* meant to be as highly tuned or robust
00148 // as a compiled regex library, but should perform well enough for a
00149 // death test, which already incurs significant overhead by launching
00150 // a child process.
00151 //
00152 // Known caveats:
00153 //
00154 // A "threadsafe" style death test obtains the path to the test
00155 // program from argv[0] and re-executes it in the sub-process. For
00156 // simplicity, the current implementation doesn't search the PATH
00157 // when launching the sub-process. This means that the user must
00158 // invoke the test program via a path that contains at least one
00159 // path separator (e.g. path/to/foo_test and
00160 // /absolute/path/to/bar_test are fine, but foo_test is not). This
00161 // is rarely a problem as people usually don't put the test binary
00162 // directory in PATH.
00163 //
00164 // FIXME: make thread-safe death tests search the PATH.
00165
00166 // Asserts that a given statement causes the program to exit, with an
00167 // integer exit status that satisfies predicate, and emitting error output
00168 // that matches regex.
00169 # define ASSERT_EXIT(statement, predicate, regex) \  

```

```

00170     GTEST_DEATH_TEST_(statement, predicate, regex, GTEST_FATAL_FAILURE_)
00171
00172 // Like ASSERT_EXIT, but continues on to successive tests in the
00173 // test case, if any:
00174 # define EXPECT_EXIT(statement, predicate, regex) \
00175     GTEST_DEATH_TEST_(statement, predicate, regex, GTEST_NONFATAL_FAILURE_)
00176
00177 // Asserts that a given statement causes the program to exit, either by
00178 // explicitly exiting with a nonzero exit code or being killed by a
00179 // signal, and emitting error output that matches regex.
00180 # define ASSERT_DEATH(statement, regex) \
00181     ASSERT_EXIT(statement, ::testing::internal::ExitedUnsuccessfully, regex)
00182
00183 // Like ASSERT_DEATH, but continues on to successive tests in the
00184 // test case, if any:
00185 # define EXPECT_DEATH(statement, regex) \
00186     EXPECT_EXIT(statement, ::testing::internal::ExitedUnsuccessfully, regex)
00187
00188 // Two predicate classes that can be used in {ASSERT,EXPECT}_EXIT*:
00189
00190 // Tests that an exit code describes a normal exit with a given exit code.
00191 class GTEST_API_ ExitedWithCode {
00192 public:
00193     explicit ExitedWithCode(int exit_code);
00194     bool operator()(int exit_status) const;
00195 private:
00196     // No implementation - assignment is unsupported.
00197     void operator=(const ExitedWithCode& other);
00198
00199     const int exit_code_;
00200 };
00201
00202 # if !GTEST_OS_WINDOWS && !GTEST_OS_FUCHSIA
00203 // Tests that an exit code describes an exit due to termination by a
00204 // given signal.
00205 // GOOGLETEST_CM0006 DO NOT DELETE
00206 class GTEST_API_ KilledBySignal {
00207 public:
00208     explicit KilledBySignal(int signum);
00209     bool operator()(int exit_status) const;
00210 private:
00211     const int signum_;
00212 };
00213 # endif // !GTEST_OS_WINDOWS
00214
00215 // EXPECT_DEBUG_DEATH asserts that the given statements die in debug mode.
00216 // The death testing framework causes this to have interesting semantics,
00217 // since the sideeffects of the call are only visible in opt mode, and not
00218 // in debug mode.
00219 //
00220 // In practice, this can be used to test functions that utilize the
00221 // LOG(DFATAL) macro using the following style:
00222 //
00223 // int DieInDebugOr12(int* sideeffect) {
00224 //     if (sideeffect) {
00225 //         *sideeffect = 12;
00226 //     }
00227 //     LOG(DFATAL) << "death";
00228 //     return 12;
00229 // }
00230 //
00231 // TEST(TestCase, TestDieOr12WorksInDgbAndOpt) {
00232 //     int sideeffect = 0;
00233 //     // Only asserts in dbg.
00234 //     EXPECT_DEBUG_DEATH(DieInDebugOr12(&sideeffect), "death");
00235 //
00236 // #ifdef NDEBUG
00237 //     // opt-mode has sideeffect visible.
00238 //     EXPECT_EQ(12, sideeffect);
00239 // #else
00240 //     // dbg-mode no visible sideeffect.
00241 //     EXPECT_EQ(0, sideeffect);
00242 // #endif
00243 // }
00244 //
00245 // This will assert that DieInDebugReturn12InOpt() crashes in debug
00246 // mode, usually due to a DCHECK or LOG(DFATAL), but returns the
00247 // appropriate fallback value (12 in this case) in opt mode. If you
00248 // need to test that a function has appropriate side-effects in opt
00249 // mode, include assertions against the side-effects. A general
00250 // pattern for this is:
00251 //
00252 // EXPECT_DEBUG_DEATH({
00253 //     // Side-effects here will have an effect after this statement in
00254 //     // opt mode, but none in debug mode.
00255 //     EXPECT_EQ(12, DieInDebugOr12(&sideeffect));
00256 // }, "death");

```

```
00257 //  
00258 # ifdef NDEBUG  
00259  
00260 # define EXPECT_DEBUG_DEATH(statement, regex) \  
00261   GTEST_EXECUTE_STATEMENT_(statement, regex)  
00262  
00263 # define ASSERT_DEBUG_DEATH(statement, regex) \  
00264   GTEST_EXECUTE_STATEMENT_(statement, regex)  
00265  
00266 # else  
00267  
00268 # define EXPECT_DEBUG_DEATH(statement, regex) \  
00269   EXPECT_DEATH(statement, regex)  
00270  
00271 # define ASSERT_DEBUG_DEATH(statement, regex) \  
00272   ASSERT_DEATH(statement, regex)  
00273  
00274 # endif // NDEBUG for EXPECT_DEBUG_DEATH  
00275 #endif // GTEST_HAS_DEATH_TEST  
00276  
00277 // This macro is used for implementing macros such as  
00278 // EXPECT_DEATH_IF_SUPPORTED and ASSERT_DEATH_IF_SUPPORTED on systems where  
00279 // death tests are not supported. Those macros must compile on such systems  
00280 // iff EXPECT_DEATH and ASSERT_DEATH compile with the same parameters on  
00281 // systems that support death tests. This allows one to write such a macro  
00282 // on a system that does not support death tests and be sure that it will  
00283 // compile on a death-test supporting system. It is exposed publicly so that  
00284 // systems that have death-tests with stricter requirements than  
00285 // GTEST_HAS_DEATH_TEST can write their own equivalent of  
00286 // EXPECT_DEATH_IF_SUPPORTED and ASSERT_DEATH_IF_SUPPORTED.  
00287 //  
00288 // Parameters:  
00289 //   statement - A statement that a macro such as EXPECT_DEATH would test  
00290 //                 for program termination. This macro has to make sure this  
00291 //                 statement is compiled but not executed, to ensure that  
00292 //                 EXPECT_DEATH_IF_SUPPORTED compiles with a certain  
00293 //                 parameter iff EXPECT_DEATH compiles with it.  
00294 //   regex      - A regex that a macro such as EXPECT_DEATH would use to test  
00295 //                 the output of statement. This parameter has to be  
00296 //                 compiled but not evaluated by this macro, to ensure that  
00297 //                 this macro only accepts expressions that a macro such as  
00298 //                 EXPECT_DEATH would accept.  
00299 //   terminator - Must be an empty statement for EXPECT_DEATH_IF_SUPPORTED  
00300 //                 and a return statement for ASSERT_DEATH_IF_SUPPORTED.  
00301 //                 This ensures that ASSERT_DEATH_IF_SUPPORTED will not  
00302 //                 compile inside functions where ASSERT_DEATH doesn't  
00303 //                 compile.  
00304 //  
00305 // The branch that has an always false condition is used to ensure that  
00306 // statement and regex are compiled (and thus syntactically correct) but  
00307 // never executed. The unreachable code macro protects the terminator  
00308 // statement from generating an 'unreachable code' warning in case  
00309 // statement unconditionally returns or throws. The Message constructor at  
00310 // the end allows the syntax of streaming additional messages into the  
00311 // macro, for compilation compatibility with EXPECT_DEATH/ASSERT_DEATH.  
00312 # define GTEST_UNSUPPORTED_DEATH_TEST(statement, regex, terminator) \  
00313   GTEST_AMBIGUOUS_ELSE_BLOCKER_ \  
00314   if (::testing::internal::AlwaysTrue()) { \  
00315     GTEST_LOG_(WARNING) \  
00316       « "Death tests are not supported on this platform.\n" \  
00317       « "Statement '" #statement "' cannot be verified."; \  
00318   } else if (::testing::internal::AlwaysFalse()) { \  
00319     ::testing::internal::RE::PartialMatch(".*", (regex)); \  
00320     GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement); \  
00321     terminator; \  
00322   } else \  
00323     ::testing::Message()  
00324  
00325 // EXPECT_DEATH_IF_SUPPORTED(statement, regex) and  
00326 // ASSERT_DEATH_IF_SUPPORTED(statement, regex) expand to real death tests if  
00327 // death tests are supported; otherwise they just issue a warning. This is  
00328 // useful when you are combining death test assertions with normal test  
00329 // assertions in one test.  
00330 #if GTEST_HAS_DEATH_TEST  
00331 # define EXPECT_DEATH_IF_SUPPORTED(statement, regex) \  
00332   EXPECT_DEATH(statement, regex)  
00333 # define ASSERT_DEATH_IF_SUPPORTED(statement, regex) \  
00334   ASSERT_DEATH(statement, regex)  
00335 #else  
00336 # define EXPECT_DEATH_IF_SUPPORTED(statement, regex) \  
00337   GTEST_UNSUPPORTED_DEATH_TEST(statement, regex, )  
00338 # define ASSERT_DEATH_IF_SUPPORTED(statement, regex) \  
00339   GTEST_UNSUPPORTED_DEATH_TEST(statement, regex, return)  
00340 #endif  
00341  
00342 } // namespace testing  
00343
```

```
00344 #endif // GTEST_INCLUDE_GTEST_GTEST_DEATH_TEST_H_
```

9.6 Dokumentacja pliku

packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-message.h

```
#include <limits>
#include "gtest/internal/gtest-port.h"
```

Komponenty

- class [testing::Message](#)

Przestrzenie nazw

- namespace [testing](#)
- namespace [testing::internal](#)

Funkcje

- std::ostream & [testing::operator<<](#) (std::ostream &os, const Message &sb)
- template<typename T> std::string [testing::internal::StreamableToString](#) (const T &streamable)

9.7 gtest-message.h

Idź do dokumentacji tego pliku.

```
00001 // Copyright 2005, Google Inc.
00002 // All rights reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //      * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //      * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 //
```

```
00031 // The Google C++ Testing and Mocking Framework (Google Test)
00032 //
00033 // This header file defines the Message class.
00034 //
00035 // IMPORTANT NOTE: Due to limitation of the C++ language, we have to
00036 // leave some internal implementation details in this header file.
00037 // They are clearly marked by comments like this:
00038 //
00039 // // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
00040 //
00041 // Such code is NOT meant to be used by a user directly, and is subject
00042 // to CHANGE WITHOUT NOTICE. Therefore DO NOT DEPEND ON IT in a user
00043 // program!
00044
00045 // GOOGLETTEST_CM0001 DO NOT DELETE
00046
00047 #ifndef GTEST_INCLUDE_GTEST_GTEST_MESSAGE_H_
00048 #define GTEST_INCLUDE_GTEST_GTEST_MESSAGE_H_
00049
00050 #include <limits>
00051
00052 #include "gtest/internal/gtest-port.h"
00053
00054 GTEST_DISABLE_MSVC_WARNINGS_PUSH_(4251 \
00055 /* class A needs to have dll-interface to be used by clients of class B */)
00056
00057 // Ensures that there is at least one operator<< in the global namespace.
00058 // See Message& operator<<(...) below for why.
00059 void operator<<(const testing::internal::Secret&, int);
00060
00061 namespace testing {
00062
00063 // The Message class works like an ostream repeater.
00064 //
00065 // Typical usage:
00066 //
00067 // 1. You stream a bunch of values to a Message object.
00068 //     It will remember the text in a stringstream.
00069 // 2. Then you stream the Message object to an ostream.
00070 //     This causes the text in the Message to be streamed
00071 //     to the ostream.
00072 //
00073 // For example;
00074 //
00075 //     testing::Message foo;
00076 //     foo << 1 << " != " << 2;
00077 //     std::cout << foo;
00078 //
00079 // will print "1 != 2".
00080 //
00081 // Message is not intended to be inherited from. In particular, its
00082 // destructor is not virtual.
00083 //
00084 // Note that stringstream behaves differently in gcc and in MSVC. You
00085 // can stream a NULL char pointer to it in the former, but not in the
00086 // latter (it causes an access violation if you do). The Message
00087 // class hides this difference by treating a NULL char pointer as
00088 // "(null)".
00089 class GTEST_API_ Message {
00090 private:
00091     // The type of basic IO manipulators (endl, ends, and flush) for
00092     // narrow streams.
00093     typedef std::ostream& (*BasicNarrowIoManip)(std::ostream&);
00094
00095 public:
00096     // Constructs an empty Message.
00097     Message();
00098
00099     // Copy constructor.
00100     Message(const Message& msg) : ss_(new ::std::stringstream) { // NOLINT
00101         *ss_ << msg.GetString();
00102     }
00103
00104     // Constructs a Message from a C-string.
00105     explicit Message(const char* str) : ss_(new ::std::stringstream) {
00106         *ss_ << str;
00107     }
00108
00109 #if GTEST_OS_SYMBIAN
00110     // Streams a value (either a pointer or not) to this object.
00111     template <typename T>
00112     inline Message& operator << (const T& value) {
00113         StreamHelper(typename internal::is_pointer<T>::type(), value);
00114         return *this;
00115     }
00116 #else
00117     // Streams a non-pointer value to this object.
00118 
```

```

00118 template <typename T>
00119 inline Message& operator <<(const T& val) {
00120     // Some libraries overload << for STL containers. These
00121     // overloads are defined in the global namespace instead of ::std.
00122     //
00123     // C++'s symbol lookup rule (i.e. Koenig lookup) says that these
00124     // overloads are visible in either the std namespace or the global
00125     // namespace, but not other namespaces, including the testing
00126     // namespace which Google Test's Message class is in.
00127     //
00128     // To allow STL containers (and other types that has a << operator
00129     // defined in the global namespace) to be used in Google Test
00130     // assertions, testing::Message must access the custom << operator
00131     // from the global namespace. With this using declaration,
00132     // overloads of << defined in the global namespace and those
00133     // visible via Koenig lookup are both exposed in this function.
00134     using ::operator <<;
00135     *ss_ << val;
00136     return *this;
00137 }
00138
00139 // Streams a pointer value to this object.
00140 //
00141 // This function is an overload of the previous one. When you
00142 // stream a pointer to a Message, this definition will be used as it
00143 // is more specialized. (The C++ Standard, section
00144 // [temp.func.order].) If you stream a non-pointer, then the
00145 // previous definition will be used.
00146 //
00147 // The reason for this overload is that streaming a NULL pointer to
00148 // ostream is undefined behavior. Depending on the compiler, you
00149 // may get "0", "(nil)", "(null)", or an access violation. To
00150 // ensure consistent result across compilers, we always treat NULL
00151 // as "(null)".
00152 template <typename T>
00153 inline Message& operator <<(T* const& pointer) { // NOLINT
00154     if (pointer == NULL) {
00155         *ss_ << "(null)";
00156     } else {
00157         *ss_ << pointer;
00158     }
00159     return *this;
00160 }
00161 #endif // GTEST_OS_SYMBIAN
00162
00163 // Since the basic IO manipulators are overloaded for both narrow
00164 // and wide streams, we have to provide this specialized definition
00165 // of operator <<, even though its body is the same as the
00166 // templated version above. Without this definition, streaming
00167 // endl or other basic IO manipulators to Message will confuse the
00168 // compiler.
00169 Message& operator <<(BasicNarrowIoManip val) {
00170     *ss_ << val;
00171     return *this;
00172 }
00173
00174 // Instead of 1/0, we want to see true/false for bool values.
00175 Message& operator <<(bool b) {
00176     return *this << (b ? "true" : "false");
00177 }
00178
00179 // These two overloads allow streaming a wide C string to a Message
00180 // using the UTF-8 encoding.
00181 Message& operator <<(const wchar_t* wide_c_str);
00182 Message& operator <<(wchar_t* wide_c_str);
00183
00184 #if GTEST_HAS_STD_WSTRING
00185     // Converts the given wide string to a narrow string using the UTF-8
00186     // encoding, and streams the result to this Message object.
00187     Message& operator <<(const ::std::wstring& wstr);
00188 #endif // GTEST_HAS_STD_WSTRING
00189
00190 #if GTEST_HAS_GLOBAL_WSTRING
00191     // Converts the given wide string to a narrow string using the UTF-8
00192     // encoding, and streams the result to this Message object.
00193     Message& operator <<(const ::wstring& wstr);
00194 #endif // GTEST_HAS_GLOBAL_WSTRING
00195
00196     // Gets the text streamed to this object so far as an std::string.
00197     // Each '\0' character in the buffer is replaced with "\\0".
00198     //
00199     // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
00200     std::string GetString() const;
00201
00202 private:
00203 #if GTEST_OS_SYMBIAN
00204     // These are needed as the Nokia Symbian Compiler cannot decide between

```

```
00205 // const T& and const T* in a function template. The Nokia compiler _can_
00206 // decide between class template specializations for T and T*, so a
00207 // tr1::type_traits-like is_pointer works, and we can overload on that.
00208 template <typename T>
00209 inline void StreamHelper(internal::true_type /*is_pointer*/, T* pointer) {
00210     if (pointer == NULL) {
00211         *ss_ << "(null)";
00212     } else {
00213         *ss_ << pointer;
00214     }
00215 }
00216 template <typename T>
00217 inline void StreamHelper(internal::false_type /*is_pointer*/,
00218                           const T& value) {
00219     // See the comments in Message& operator <<(const T&) above for why
00220     // we need this using statement.
00221     using ::operator <<;
00222     *ss_ << value;
00223 }
00224 #endif // GTEST_OS_SYMBIAN
00225
00226 // We'll hold the text streamed to this object here.
00227 const internal::scoped_ptr<::std::stringstream> ss_;
00228
00229 // We declare (but don't implement) this to prevent the compiler
00230 // from implementing the assignment operator.
00231 void operator=(const Message&);
00232 };
00233
00234 // Streams a Message to an ostream.
00235 inline std::ostream& operator <<(std::ostream& os, const Message& sb) {
00236     return os << sb.GetString();
00237 }
00238
00239 namespace internal {
00240
00241 // Converts a streamable value to an std::string. A NULL pointer is
00242 // converted to "(null)". When the input value is a ::string,
00243 // ::std::string, ::wstring, or ::std::wstring object, each NUL
00244 // character in it is replaced with "\\0".
00245 template <typename T>
00246 std::string StreamableToString(const T& streamable) {
00247     return (Message() << streamable).GetString();
00248 }
00249
00250 } // namespace internal
00251 } // namespace testing
00252
00253 GTEST_DISABLE_MSC_WARNINGS_POP_() // 4251
00254
00255 #endif // GTEST_INCLUDE_GTEST_MESSAGE_H_
```

9.8 Dokumentacja pliku packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-param-test.h

```
#include "gtest/internal/gtest-port.h"
#include <utility>
#include "gtest/internal/gtest-internal.h"
#include "gtest/internal/gtest-param-util.h"
#include "gtest/internal/gtest-param-util-generated.h"
```

Przestrzenie nazw

- namespace **testing**

Definicje

- #define **TEST_P**(test_case_name, test_name)
- #define **INSTANTIATE_TEST_CASE_P**(prefix, test_case_name, generator, ...)

Funkcje

- template<typename T, typename IncrementT>
`internal::ParamGenerator< T > testing::Range (T start, T end, IncrementT step)`
- template<typename T>
`internal::ParamGenerator< T > testing::Range (T start, T end)`
- template<typename ForwardIterator>
`internal::ParamGenerator< typename ::testing::internal::IteratorTraits< ForwardIterator >::value_type > testing::ValuesIn (ForwardIterator begin, ForwardIterator end)`
- template<typename T, size_t N>
`internal::ParamGenerator< T > testing::ValuesIn (const T(&array)[N])`
- template<class Container>
`internal::ParamGenerator< typename Container::value_type > testing::ValuesIn (const Container &container)`
- template<typename T1>
`internal::ValueArray1< T1 > testing::Values (T1 v1)`
- template<typename T1, typename T2>
`internal::ValueArray2< T1, T2 > testing::Values (T1 v1, T2 v2)`
- template<typename T1, typename T2, typename T3>
`internal::ValueArray3< T1, T2, T3 > testing::Values (T1 v1, T2 v2, T3 v3)`
- template<typename T1, typename T2, typename T3, typename T4>
`internal::ValueArray4< T1, T2, T3, T4 > testing::Values (T1 v1, T2 v2, T3 v3, T4 v4)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5>
`internal::ValueArray5< T1, T2, T3, T4, T5 > testing::Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>
`internal::ValueArray6< T1, T2, T3, T4, T5, T6 > testing::Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7>
`internal::ValueArray7< T1, T2, T3, T4, T5, T6, T7 > testing::Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T8>
`internal::ValueArray8< T1, T2, T3, T4, T5, T6, T7, T8 > testing::Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9>
`internal::ValueArray9< T1, T2, T3, T4, T5, T6, T7, T8, T9 > testing::Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10>
`internal::ValueArray10< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 > testing::Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11>
`internal::ValueArray11< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T11 > testing::Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T11 v11)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12>
`internal::ValueArray12< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T11, T12 > testing::Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T12 v12)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13>
`internal::ValueArray13< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T11, T12, T13 > testing::Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13)`
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14>
`internal::ValueArray14< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14 > testing::Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14)`

v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24)

v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31)

v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37)

```
name T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26,
typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43>
internal::ValueArray43< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18,
T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
T40, T41, T42, T43 > testing::Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10
v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21,
T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33
v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43)
```

- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename
T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename
T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename
T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename
T43, typename T44>
internal::ValueArray44< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18,
T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
T40, T41, T42, T43, T44 > **testing::Values** (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21
v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32,
T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44
v44)
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename
T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename
T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename
T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename
T43, typename T44, typename T45>
internal::ValueArray45< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18,
T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
T40, T41, T42, T43, T44, T45 > **testing::Values** (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9
v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20,
T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32
v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43,
T44 v44, T45 v45)
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename
T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename
T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename
T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename
T43, typename T44, typename T45, typename T46>
internal::ValueArray46< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18,
T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
T40, T41, T42, T43, T44, T45, T46 > **testing::Values** (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8
v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20
v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32
v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43,
T44 v44, T45 v45, T46 v46)
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename
T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename
T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename
T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename
T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename
T43, typename T44, typename T45, typename T46, typename T47>
internal::ValueArray47< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18,
T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
T40, T41, T42, T43, T44, T45, T46, T47 > **testing::Values** (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7,
T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19,

T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47)

- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48>
internal::ValueArray48< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48 > **testing::Values** (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47, T48 v48)
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48, typename T49>
internal::ValueArray49< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49 > **testing::Values** (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47, T48 v48, T49 v49)
- template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14, typename T15, typename T16, typename T17, typename T18, typename T19, typename T20, typename T21, typename T22, typename T23, typename T24, typename T25, typename T26, typename T27, typename T28, typename T29, typename T30, typename T31, typename T32, typename T33, typename T34, typename T35, typename T36, typename T37, typename T38, typename T39, typename T40, typename T41, typename T42, typename T43, typename T44, typename T45, typename T46, typename T47, typename T48, typename T49, typename T50>
internal::ValueArray50< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45, T46, T47, T48, T49, T50 > **testing::Values** (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47, T48 v48, T49 v49, T50 v50)
- internal::ParamGenerator**< bool > **testing::Bool** ()

9.8.1 Dokumentacja definicji

9.8.1.1 INSTANTIATE_TEST_CASE_P

```
#define INSTANTIATE_TEST_CASE_P (
    prefix,
    test_case_name,
    generator,
    ...)
```

Wartość:

```

static ::testing::internal::ParamGenerator<test_case_name::ParamType> \
    gtest_#__prefix##test_case_name##_EvalGenerator_() { return generator; } \
static ::std::string gtest_#__prefix##test_case_name##_EvalGenerateName_( \
    const ::testing::TestParamInfo<test_case_name::ParamType>& info) { \
    return ::testing::internal::GetParamNameGen<test_case_name::ParamType> \
        (__VA_ARGS__)(info); \
} \
static int gtest_#__prefix##test_case_name##_dummy_ GTEST_ATTRIBUTE_UNUSED_ = \
    ::testing::UnitTest::GetInstance()->parameterized_test_registry(). \
    GetTestCasePatternHolder<test_case_name>(\
        #test_case_name, \
        ::testing::internal::CodeLocation(\
            __FILE__, __LINE__))->AddTestCaseInstantiation(\
                #prefix, \
                &gtest_#__prefix##test_case_name##_EvalGenerator_, \
                &gtest_#__prefix##test_case_name##_EvalGenerateName_, \
                __FILE__, __LINE__)

```

9.8.1.2 TEST_P

```
#define TEST_P( \
    test_case_name, \
    test_name)
```

Wartość:

```

class GTEST_TEST_CLASS_NAME_(test_case_name, test_name) \
    : public test_case_name { \
public: \
    GTEST_TEST_CLASS_NAME_(test_case_name, test_name)() {} \
    virtual void TestBody(); \
private: \
    static int AddToRegistry() { \
        ::testing::UnitTest::GetInstance()->parameterized_test_registry(). \
        GetTestCasePatternHolder<test_case_name>(\
            #test_case_name, \
            ::testing::internal::CodeLocation(\
                __FILE__, __LINE__)->AddTestPattern(\
                    GTEST_STRINGIFY_(test_case_name), \
                    GTEST_STRINGIFY_(test_name), \
                    new ::testing::internal::TestMetaFactory< \
                        GTEST_TEST_CLASS_NAME_(< \
                            test_case_name, test_name)>()); \
        return 0; \
    } \
    static int gtest_registering_dummy_ GTEST_ATTRIBUTE_UNUSED_; \
    GTEST_DISALLOW_COPY_AND_ASSIGN_( \
        GTEST_TEST_CLASS_NAME_(test_case_name, test_name)); \
}; \
int GTEST_TEST_CLASS_NAME_(test_case_name, \
    test_name)::gtest_registering_dummy_ = \
    GTEST_TEST_CLASS_NAME_(test_case_name, test_name)::AddToRegistry(); \
void GTEST_TEST_CLASS_NAME_(test_case_name, test_name)::TestBody()

```

9.9 gtest-param-test.h

Idź do dokumentacji tego pliku.

```

00001 // This file was GENERATED by command:
00002 //     pump.py gtest-param-test.h.pump
00003 // DO NOT EDIT BY HAND!!!
00004
00005 // Copyright 2008, Google Inc.
00006 // All rights reserved.
00007 //
00008 // Redistribution and use in source and binary forms, with or without
00009 // modification, are permitted provided that the following conditions are
00010 // met:
00011 //
00012 //     * Redistributions of source code must retain the above copyright
00013 // notice, this list of conditions and the following disclaimer.
00014 //     * Redistributions in binary form must reproduce the above
00015 // copyright notice, this list of conditions and the following disclaimer
00016 // in the documentation and/or other materials provided with the
00017 // distribution.
00018 //     * Neither the name of Google Inc. nor the names of its

```

```

00019 // contributors may be used to endorse or promote products derived from
00020 // this software without specific prior written permission.
00021 //
00022 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00023 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00024 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00025 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00026 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00027 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00028 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00029 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00030 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00031 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00032 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00033 //
00034 // Macros and functions for implementing parameterized tests
00035 // in Google C++ Testing and Mocking Framework (Google Test)
00036 //
00037 // This file is generated by a SCRIPT. DO NOT EDIT BY HAND!
00038 //
00039 // GOOGLETEST_CM0001 DO NOT DELETE
00040 #ifndef GTEST_INCLUDE_GTEST_GTEST_PARAM_TEST_H_
00041 #define GTEST_INCLUDE_GTEST_GTEST_PARAM_TEST_H_
00042
00043
00044 // Value-parameterized tests allow you to test your code with different
00045 // parameters without writing multiple copies of the same test.
00046 //
00047 // Here is how you use value-parameterized tests:
00048
00049 #if 0
00050
00051 // To write value-parameterized tests, first you should define a fixture
00052 // class. It is usually derived from testing::TestWithParam<T> (see below for
00053 // another inheritance scheme that's sometimes useful in more complicated
00054 // class hierarchies), where the type of your parameter values.
00055 // TestWithParam<T> is itself derived from testing::Test. T can be any
00056 // copyable type. If it's a raw pointer, you are responsible for managing the
00057 // lifespan of the pointed values.
00058
00059 class FooTest : public ::testing::TestWithParam<const char*> {
00060   // You can implement all the usual class fixture members here.
00061 };
00062
00063 // Then, use the TEST_P macro to define as many parameterized tests
00064 // for this fixture as you want. The _P suffix is for "parameterized"
00065 // or "pattern", whichever you prefer to think.
00066
00067 TEST_P(FooTest, DoesBlah) {
00068   // Inside a test, access the test parameter with the GetParam() method
00069   // of the TestWithParam<T> class:
00070   EXPECT_TRUE(foo.Blah(GetParam()));
00071   ...
00072 }
00073
00074 TEST_P(FooTest, HasBlahBlah) {
00075   ...
00076 }
00077
00078 // Finally, you can use INSTANTIATE_TEST_CASE_P to instantiate the test
00079 // case with any set of parameters you want. Google Test defines a number
00080 // of functions for generating test parameters. They return what we call
00081 // (surprise!) parameter generators. Here is a summary of them, which
00082 // are all in the testing namespace:
00083 //
00084 //
00085 // Range(begin, end [, step]) - Yields values {begin, begin+step,
00086 //                                         begin+step+step, ...}. The values do not
00087 //                                         include end. step defaults to 1.
00088 // Values(v1, v2, ..., vN)    - Yields values {v1, v2, ..., vN}.
00089 // ValuesIn(container)      - Yields values from a C-style array, an STL
00090 //                               container, or an iterator range [begin, end).
00091 // Bool()                   - Yields sequence {false, true}.
00092 // Combine(g1, g2, ..., gN) - Yields all combinations (the Cartesian product
00093 //                               for the math savvy) of the values generated
00094 //                                         by the N generators.
00095 //
00096 // For more details, see comments at the definitions of these functions below
00097 // in this file.
00098 //
00099 // The following statement will instantiate tests from the FooTest test case
00100 // each with parameter values "meeny", "miny", and "moe".
00101
00102 INSTANTIATE_TEST_CASE_P(InstantiationName,
00103                           FooTest,
00104                           Values("meeny", "miny", "moe"));
00105

```

```
00106 // To distinguish different instances of the pattern, (yes, you
00107 // can instantiate it more than once) the first argument to the
00108 // INSTANTIATE_TEST_CASE_P macro is a prefix that will be added to the
00109 // actual test case name. Remember to pick unique prefixes for different
00110 // instantiations. The tests from the instantiation above will have
00111 // these names:
00112 //
00113 //     * InstantiationName/FooTest.DoesBlah/0 for "meeny"
00114 //     * InstantiationName/FooTest.DoesBlah/1 for "miny"
00115 //     * InstantiationName/FooTest.DoesBlah/2 for "moe"
00116 //     * InstantiationName/FooTest.HasBlahBlah/0 for "meeny"
00117 //     * InstantiationName/FooTest.HasBlahBlah/1 for "miny"
00118 //     * InstantiationName/FooTest.HasBlahBlah/2 for "moe"
00119 //
00120 // You can use these names in --gtest_filter.
00121 //
00122 // This statement will instantiate all tests from FooTest again, each
00123 // with parameter values "cat" and "dog":
00124
00125 const char* pets[] = {"cat", "dog"};
00126 INSTANTIATE_TEST_CASE_P(AnotherInstantiationName, FooTest, ValuesIn(pets));
00127
00128 // The tests from the instantiation above will have these names:
00129 //
00130 //     * AnotherInstantiationName/FooTest.DoesBlah/0 for "cat"
00131 //     * AnotherInstantiationName/FooTest.DoesBlah/1 for "dog"
00132 //     * AnotherInstantiationName/FooTest.HasBlahBlah/0 for "cat"
00133 //     * AnotherInstantiationName/FooTest.HasBlahBlah/1 for "dog"
00134 //
00135 // Please note that INSTANTIATE_TEST_CASE_P will instantiate all tests
00136 // in the given test case, whether their definitions come before or
00137 // AFTER the INSTANTIATE_TEST_CASE_P statement.
00138 //
00139 // Please also note that generator expressions (including parameters to the
00140 // generators) are evaluated in InitGoogleTest(), after main() has started.
00141 // This allows the user on one hand, to adjust generator parameters in order
00142 // to dynamically determine a set of tests to run and on the other hand,
00143 // give the user a chance to inspect the generated tests with Google Test
00144 // reflection API before RUN_ALL_TESTS() is executed.
00145 //
00146 // You can see samples/sample7_unittest.cc and samples/sample8_unittest.cc
00147 // for more examples.
00148 //
00149 // In the future, we plan to publish the API for defining new parameter
00150 // generators. But for now this interface remains part of the internal
00151 // implementation and is subject to change.
00152 //
00153 //
00154 // A parameterized test fixture must be derived from testing::Test and from
00155 // testing::WithParamInterface<T>, where T is the type of the parameter
00156 // values. Inheriting from TestWithParam<T> satisfies that requirement because
00157 // TestWithParam<T> inherits from both Test and WithParamInterface. In more
00158 // complicated hierarchies, however, it is occasionally useful to inherit
00159 // separately from Test and WithParamInterface. For example:
00160
00161 class BaseTest : public ::testing::Test {
00162     // You can inherit all the usual members for a non-parameterized test
00163     // fixture here.
00164 };
00165
00166 class DerivedTest : public BaseTest, public ::testing::WithParamInterface<int> {
00167     // The usual test fixture members go here too.
00168 };
00169
00170 TEST_F(BaseTest, HasFoo) {
00171     // This is an ordinary non-parameterized test.
00172 }
00173
00174 TEST_P(DerivedTest, DoesBlah) {
00175     // GetParam works just the same here as if you inherit from TestWithParam.
00176     EXPECT_TRUE(foo.Blah(GetParam()));
00177 }
00178
00179 #endif // 0
00180
00181 #include "gtest/internal/gtest-port.h"
00182
00183 #if !GTEST_OS_SYMBIAN
00184 # include <utility>
00185 #endif
00186
00187 #include "gtest/internal/gtest-internal.h"
00188 #include "gtest/internal/gtest-param-util.h"
00189 #include "gtest/internal/gtest-param-util-generated.h"
00190
00191 namespace testing {
00192
```

```

00193 // Functions producing parameter generators.
00194 //
00195 // Google Test uses these generators to produce parameters for value-
00196 // parameterized tests. When a parameterized test case is instantiated
00197 // with a particular generator, Google Test creates and runs tests
00198 // for each element in the sequence produced by the generator.
00199 //
00200 // In the following sample, tests from test case FooTest are instantiated
00201 // each three times with parameter values 3, 5, and 8:
00202 //
00203 // class FooTest : public TestWithParam<int> { ... };
00204 //
00205 // TEST_P(FooTest, TestThis) {
00206 // }
00207 // TEST_P(FooTest, TestThat) {
00208 // }
00209 // INSTANTIATE_TEST_CASE_P(TestSequence, FooTest, Values(3, 5, 8));
00210 //
00211
00212 // Range() returns generators providing sequences of values in a range.
00213 //
00214 // Synopsis:
00215 // Range(start, end)
00216 //   - returns a generator producing a sequence of values {start, start+1,
00217 //     start+2, ...}.
00218 // Range(start, end, step)
00219 //   - returns a generator producing a sequence of values {start, start+step,
00220 //     start+step+step, ...}.
00221 // Notes:
00222 //   * The generated sequences never include end. For example, Range(1, 5)
00223 //     returns a generator producing a sequence {1, 2, 3, 4}. Range(1, 9, 2)
00224 //     returns a generator producing {1, 3, 5, 7}.
00225 //   * start and end must have the same type. That type may be any integral or
00226 //     floating-point type or a user defined type satisfying these conditions:
00227 //     * It must be assignable (have operator=() defined).
00228 //     * It must have operator+() (operator+(int-compatible type) for
00229 //       two-operand version).
00230 //     * It must have operator<() defined.
00231 //     Elements in the resulting sequences will also have that type.
00232 //   * Condition start < end must be satisfied in order for resulting sequences
00233 //     to contain any elements.
00234 //
00235 template <typename T, typename IncrementT>
00236 internal::ParamGenerator<T> Range(T start, T end, IncrementT step) {
00237     return internal::ParamGenerator<T>(
00238         new internal::RangeGenerator<T, IncrementT>(start, end, step));
00239 }
00240
00241 template <typename T>
00242 internal::ParamGenerator<T> Range(T start, T end) {
00243     return Range(start, end, 1);
00244 }
00245
00246 // ValuesIn() function allows generation of tests with parameters coming from
00247 // a container.
00248 //
00249 // Synopsis:
00250 // ValuesIn(const T (&array)[N])
00251 //   - returns a generator producing sequences with elements from
00252 //     a C-style array.
00253 // ValuesIn(const Container& container)
00254 //   - returns a generator producing sequences with elements from
00255 //     an STL-style container.
00256 // ValuesIn(Iterator begin, Iterator end)
00257 //   - returns a generator producing sequences with elements from
00258 //     a range [begin, end) defined by a pair of STL-style iterators. These
00259 //     iterators can also be plain C pointers.
00260 //
00261 // Please note that ValuesIn copies the values from the containers
00262 // passed in and keeps them to generate tests in RUN_ALL_TESTS().
00263 //
00264 // Examples:
00265 //
00266 // This instantiates tests from test case StringTest
00267 // each with C-string values of "foo", "bar", and "baz":
00268 //
00269 // const char* strings[] = {"foo", "bar", "baz"};
00270 // INSTANTIATE_TEST_CASE_P(StringSequence, StringTest, ValuesIn(strings));
00271 //
00272 // This instantiates tests from test case StlStringTest
00273 // each with STL strings with values "a" and "b":
00274 //
00275 // ::std::vector< ::std::string> GetParameterStrings() {
00276 //     ::std::vector< ::std::string> v;
00277 //     v.push_back("a");
00278 //     v.push_back("b");
00279 //     return v;

```

```
00280 // }
00281 //
00282 // INSTANTIATE_TEST_CASE_P(CharSequence,
00283 //                           StlStringTest,
00284 //                           ValuesIn(GetParameterStrings()));
00285 //
00286 //
00287 // This will also instantiate tests from CharTest
00288 // each with parameter values 'a' and 'b':
00289 //
00290 // ::std::list<char> GetParameterChars() {
00291 //   ::std::list<char> list;
00292 //   list.push_back('a');
00293 //   list.push_back('b');
00294 //   return list;
00295 // }
00296 // ::std::list<char> l = GetParameterChars();
00297 // INSTANTIATE_TEST_CASE_P(CharSequence2,
00298 //                           CharTest,
00299 //                           ValuesIn(l.begin(), l.end()));
00300 //
00301 template <typename ForwardIterator>
00302 internal::ParamGenerator<
00303   typename ::testing::internal::IteratorTraits<ForwardIterator>::value_type>
00304 ValuesIn(ForwardIterator begin, ForwardIterator end) {
00305   typedef typename ::testing::internal::IteratorTraits<ForwardIterator>
00306   ::value_type ParamType;
00307   return internal::ParamGenerator<ParamType>(
00308     new internal::ValuesInIteratorRangeGenerator<ParamType>(begin, end));
00309 }
00310
00311 template <typename T, size_t N>
00312 internal::ParamGenerator<T> ValuesIn(const T (&array)[N]) {
00313   return ValuesIn(array, array + N);
00314 }
00315
00316 template <class Container>
00317 internal::ParamGenerator<typename Container::value_type> ValuesIn(
00318   const Container& container) {
00319   return ValuesIn(container.begin(), container.end());
00320 }
00321
00322 // Values() allows generating tests from explicitly specified list of
00323 // parameters.
00324 //
00325 // Synopsis:
00326 // Values(T v1, T v2, ..., T vN)
00327 //   - returns a generator producing sequences with elements v1, v2, ..., vN.
00328 //
00329 // For example, this instantiates tests from test case BarTest each
00330 // with values "one", "two", and "three":
00331 //
00332 // INSTANTIATE_TEST_CASE_P(NumSequence, BarTest, Values("one", "two", "three"));
00333 //
00334 // This instantiates tests from test case BazTest each with values 1, 2, 3.5.
00335 // The exact type of values will depend on the type of parameter in BazTest.
00336 //
00337 // INSTANTIATE_TEST_CASE_P(FloatingNumbers, BazTest, Values(1, 2, 3.5));
00338 //
00339 // Currently, Values() supports from 1 to 50 parameters.
00340 //
00341 template <typename T1>
00342 internal::ValueArray1<T1> Values(T1 v1) {
00343   return internal::ValueArray1<T1>(v1);
00344 }
00345
00346 template <typename T1, typename T2>
00347 internal::ValueArray2<T1, T2> Values(T1 v1, T2 v2) {
00348   return internal::ValueArray2<T1, T2>(v1, v2);
00349 }
00350
00351 template <typename T1, typename T2, typename T3>
00352 internal::ValueArray3<T1, T2, T3> Values(T1 v1, T2 v2, T3 v3) {
00353   return internal::ValueArray3<T1, T2, T3>(v1, v2, v3);
00354 }
00355
00356 template <typename T1, typename T2, typename T3, typename T4>
00357 internal::ValueArray4<T1, T2, T3, T4> Values(T1 v1, T2 v2, T3 v3, T4 v4) {
00358   return internal::ValueArray4<T1, T2, T3, T4>(v1, v2, v3, v4);
00359 }
00360
00361 template <typename T1, typename T2, typename T3, typename T4, typename T5>
00362 internal::ValueArray5<T1, T2, T3, T4, T5> Values(T1 v1, T2 v2, T3 v3, T4 v4,
00363   T5 v5) {
00364   return internal::ValueArray5<T1, T2, T3, T4, T5>(v1, v2, v3, v4, v5);
00365 }
00366
```

```

00367 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00368     typename T6>
00369 internal::ValueArray6<T1, T2, T3, T4, T5, T6> Values(T1 v1, T2 v2, T3 v3,
00370     T4 v4, T5 v5, T6 v6) {
00371     return internal::ValueArray6<T1, T2, T3, T4, T5, T6>(v1, v2, v3, v4, v5, v6);
00372 }
00373
00374 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00375     typename T6, typename T7>
00376 internal::ValueArray7<T1, T2, T3, T4, T5, T6, T7> Values(T1 v1, T2 v2, T3 v3,
00377     T4 v4, T5 v5, T6 v6, T7 v7) {
00378     return internal::ValueArray7<T1, T2, T3, T4, T5, T6, T7>(v1, v2, v3, v4, v5,
00379         v6, v7);
00380 }
00381
00382 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00383     typename T6, typename T7, typename T8>
00384 internal::ValueArray8<T1, T2, T3, T4, T5, T6, T7, T8> Values(T1 v1, T2 v2,
00385     T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8) {
00386     return internal::ValueArray8<T1, T2, T3, T4, T5, T6, T7, T8>(v1, v2, v3, v4,
00387         v5, v6, v7, v8);
00388 }
00389
00390 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00391     typename T6, typename T7, typename T8, typename T9>
00392 internal::ValueArray9<T1, T2, T3, T4, T5, T6, T7, T8, T9> Values(T1 v1, T2 v2,
00393     T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9) {
00394     return internal::ValueArray9<T1, T2, T3, T4, T5, T6, T7, T8, T9>(v1, v2, v3,
00395         v4, v5, v6, v7, v8, v9);
00396 }
00397
00398 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00399     typename T6, typename T7, typename T8, typename T9, typename T10>
00400 internal::ValueArray10<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10> Values(T1 v1,
00401     T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10) {
00402     return internal::ValueArray10<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10>(v1,
00403         v2, v3, v4, v5, v6, v7, v8, v9, v10);
00404 }
00405
00406 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00407     typename T6, typename T7, typename T8, typename T9, typename T10,
00408     typename T11>
00409 internal::ValueArray11<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10,
00410     T11> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00411     T10 v10, T11 v11) {
00412     return internal::ValueArray11<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10,
00413         T11>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11);
00414 }
00415
00416 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00417     typename T6, typename T7, typename T8, typename T9, typename T10,
00418     typename T11, typename T12>
00419 internal::ValueArray12<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00420     T12> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00421     T10 v10, T11 v11, T12 v12) {
00422     return internal::ValueArray12<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00423         T12>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12);
00424 }
00425
00426 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00427     typename T6, typename T7, typename T8, typename T9, typename T10,
00428     typename T11, typename T12, typename T13>
00429 internal::ValueArray13<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
00430     T13> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00431     T10 v10, T11 v11, T12 v12, T13 v13) {
00432     return internal::ValueArray13<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00433         T12, T13>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13);
00434 }
00435
00436 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00437     typename T6, typename T7, typename T8, typename T9, typename T10,
00438     typename T11, typename T12, typename T13, typename T14>
00439 internal::ValueArray14<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00440     T14> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00441     T10 v10, T11 v11, T12 v12, T13 v13, T14 v14) {
00442     return internal::ValueArray14<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00443         T12, T13, T14>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13,
00444         v14);
00445 }
00446
00447 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00448     typename T6, typename T7, typename T8, typename T9, typename T10,
00449     typename T11, typename T12, typename T13, typename T14, typename T15>
00450 internal::ValueArray15<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00451     T14, T15> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8,
00452     T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15) {
00453     return internal::ValueArray15<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,

```

```

00454     T12, T13, T14, T15>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12,
00455     v13, v14, v15);
00456 }
00457
00458 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00459     typename T6, typename T7, typename T8, typename T9, typename T10,
00460     typename T11, typename T12, typename T13, typename T14, typename T15,
00461     typename T16>
00462 internal::ValueArray16<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00463     T14, T15, T16> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7,
00464     T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15,
00465     T16 v16) {
00466     return internal::ValueArray16<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00467     T12, T13, T14, T15, T16>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11,
00468     v12, v13, v14, v15, v16);
00469 }
00470
00471 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00472     typename T6, typename T7, typename T8, typename T9, typename T10,
00473     typename T11, typename T12, typename T13, typename T14, typename T15,
00474     typename T16, typename T17>
00475 internal::ValueArray17<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00476     T14, T15, T16, T17> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7,
00477     T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15,
00478     T16 v16, T17 v17) {
00479     return internal::ValueArray17<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00480     T12, T13, T14, T15, T16, T17>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10,
00481     v11, v12, v13, v14, v15, v16, v17);
00482 }
00483
00484 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00485     typename T6, typename T7, typename T8, typename T9, typename T10,
00486     typename T11, typename T12, typename T13, typename T14, typename T15,
00487     typename T16, typename T17, typename T18>
00488 internal::ValueArray18<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00489     T14, T15, T16, T17, T18> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6,
00490     T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15,
00491     T16 v16, T17 v17, T18 v18) {
00492     return internal::ValueArray18<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00493     T12, T13, T14, T15, T16, T17, T18>(v1, v2, v3, v4, v5, v6, v7, v8, v9,
00494     v10, v11, v12, v13, v14, v15, v16, v17, v18);
00495 }
00496
00497 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00498     typename T6, typename T7, typename T8, typename T9, typename T10,
00499     typename T11, typename T12, typename T13, typename T14, typename T15,
00500     typename T16, typename T17, typename T18, typename T19>
00501 internal::ValueArray19<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00502     T14, T15, T16, T17, T18, T19> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5,
00503     T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14,
00504     T15 v15, T16 v16, T17 v17, T18 v18, T19 v19) {
00505     return internal::ValueArray19<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00506     T12, T13, T14, T15, T16, T17, T18, T19>(v1, v2, v3, v4, v5, v6, v7, v8,
00507     v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19);
00508 }
00509
00510 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00511     typename T6, typename T7, typename T8, typename T9, typename T10,
00512     typename T11, typename T12, typename T13, typename T14, typename T15,
00513     typename T16, typename T17, typename T18, typename T19, typename T20>
00514 internal::ValueArray20<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00515     T14, T15, T16, T17, T18, T19, T20> Values(T1 v1, T2 v2, T3 v3, T4 v4,
00516     T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13,
00517     T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20) {
00518     return internal::ValueArray20<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00519     T12, T13, T14, T15, T16, T17, T18, T19, T20>(v1, v2, v3, v4, v5, v6, v7,
00520     v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20);
00521 }
00522
00523 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00524     typename T6, typename T7, typename T8, typename T9, typename T10,
00525     typename T11, typename T12, typename T13, typename T14, typename T15,
00526     typename T16, typename T17, typename T18, typename T19, typename T20,
00527     typename T21>
00528 internal::ValueArray21<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00529     T14, T15, T16, T17, T18, T19, T20, T21> Values(T1 v1, T2 v2, T3 v3, T4 v4,
00530     T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13,
00531     T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21) {
00532     return internal::ValueArray21<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00533     T12, T13, T14, T15, T16, T17, T18, T19, T20, T21>(v1, v2, v3, v4, v5, v6,
00534     v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21);
00535 }
00536
00537 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00538     typename T6, typename T7, typename T8, typename T9, typename T10,
00539     typename T11, typename T12, typename T13, typename T14, typename T15,
00540     typename T16, typename T17, typename T18, typename T19, typename T20,

```

```

00541     typename T21, typename T22>
00542 internal::ValueArray22<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00543     T14, T15, T16, T17, T18, T19, T20, T21, T22> Values(T1 v1, T2 v2, T3 v3,
00544     T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12,
00545     T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20,
00546     T21 v21, T22 v22) {
00547     return internal::ValueArray22<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00548         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22>(v1, v2, v3, v4,
00549         v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19,
00550         v20, v21, v22);
00551 }
00552
00553 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00554     typename T6, typename T7, typename T8, typename T9, typename T10,
00555     typename T11, typename T12, typename T13, typename T14, typename T15,
00556     typename T16, typename T17, typename T18, typename T19, typename T20,
00557     typename T21, typename T22, typename T23>
00558 internal::ValueArray23<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00559     T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23> Values(T1 v1, T2 v2,
00560     T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12,
00561     T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20,
00562     T21 v21, T22 v22, T23 v23) {
00563     return internal::ValueArray23<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00564         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23>(v1, v2, v3,
00565         v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19,
00566         v20, v21, v22, v23);
00567 }
00568
00569 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00570     typename T6, typename T7, typename T8, typename T9, typename T10,
00571     typename T11, typename T12, typename T13, typename T14, typename T15,
00572     typename T16, typename T17, typename T18, typename T19, typename T20,
00573     typename T21, typename T22, typename T23, typename T24>
00574 internal::ValueArray24<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00575     T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24> Values(T1 v1, T2 v2,
00576     T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12,
00577     T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20,
00578     T21 v21, T22 v22, T23 v23, T24 v24) {
00579     return internal::ValueArray24<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00580         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24>(v1, v2,
00581         v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18,
00582         v19, v20, v21, v22, v23, v24);
00583 }
00584
00585 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00586     typename T6, typename T7, typename T8, typename T9, typename T10,
00587     typename T11, typename T12, typename T13, typename T14, typename T15,
00588     typename T16, typename T17, typename T18, typename T19, typename T20,
00589     typename T21, typename T22, typename T23, typename T24, typename T25>
00590 internal::ValueArray25<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00591     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25> Values(T1 v1,
00592     T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11,
00593     T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19,
00594     T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25) {
00595     return internal::ValueArray25<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00596         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25>(v1,
00597         v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17,
00598         v18, v19, v20, v21, v22, v23, v24, v25);
00599 }
00600
00601 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00602     typename T6, typename T7, typename T8, typename T9, typename T10,
00603     typename T11, typename T12, typename T13, typename T14, typename T15,
00604     typename T16, typename T17, typename T18, typename T19, typename T20,
00605     typename T21, typename T22, typename T23, typename T24, typename T25,
00606     typename T26>
00607 internal::ValueArray26<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00608     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00609     T26> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00610     T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
00611     T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
00612     T26 v26) {
00613     return internal::ValueArray26<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00614         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00615         T26>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15,
00616         v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26);
00617 }
00618
00619 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00620     typename T6, typename T7, typename T8, typename T9, typename T10,
00621     typename T11, typename T12, typename T13, typename T14, typename T15,
00622     typename T16, typename T17, typename T18, typename T19, typename T20,
00623     typename T21, typename T22, typename T23, typename T24, typename T25,
00624     typename T26, typename T27>
00625 internal::ValueArray27<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00626     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
00627     T27> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
```

```

00628     T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
00629     T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
00630     T26 v26, T27 v27) {
00631     return internal::ValueArray27<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00632         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00633         T26, T27>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14,
00634         v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26, v27);
00635 }
00636
00637 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00638     typename T6, typename T7, typename T8, typename T9, typename T10,
00639     typename T11, typename T12, typename T13, typename T14, typename T15,
00640     typename T16, typename T17, typename T18, typename T19, typename T20,
00641     typename T21, typename T22, typename T23, typename T24, typename T25,
00642     typename T26, typename T27, typename T28>
00643 internal::ValueArray28<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00644     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
00645     T28> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00646     T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
00647     T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
00648     T26 v26, T27 v27, T28 v28) {
00649     return internal::ValueArray28<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00650         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00651         T26, T27, T28>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13,
00652         v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26, v27,
00653         v28);
00654 }
00655
00656 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00657     typename T6, typename T7, typename T8, typename T9, typename T10,
00658     typename T11, typename T12, typename T13, typename T14, typename T15,
00659     typename T16, typename T17, typename T18, typename T19, typename T20,
00660     typename T21, typename T22, typename T23, typename T24, typename T25,
00661     typename T26, typename T27, typename T28, typename T29>
00662 internal::ValueArray29<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00663     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00664     T29> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00665     T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
00666     T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
00667     T26 v26, T27 v27, T28 v28, T29 v29) {
00668     return internal::ValueArray29<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00669         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00670         T26, T27, T28, T29>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12,
00671         v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26,
00672         v27, v28, v29);
00673 }
00674
00675 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00676     typename T6, typename T7, typename T8, typename T9, typename T10,
00677     typename T11, typename T12, typename T13, typename T14, typename T15,
00678     typename T16, typename T17, typename T18, typename T19, typename T20,
00679     typename T21, typename T22, typename T23, typename T24, typename T25,
00680     typename T26, typename T27, typename T28, typename T29, typename T30>
00681 internal::ValueArray30<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00682     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00683     T29, T30> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8,
00684     T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16,
00685     T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24,
00686     T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30) {
00687     return internal::ValueArray30<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00688         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00689         T26, T27, T28, T29, T30>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11,
00690         v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25,
00691         v26, v27, v28, v29, v30);
00692 }
00693
00694 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00695     typename T6, typename T7, typename T8, typename T9, typename T10,
00696     typename T11, typename T12, typename T13, typename T14, typename T15,
00697     typename T16, typename T17, typename T18, typename T19, typename T20,
00698     typename T21, typename T22, typename T23, typename T24, typename T25,
00699     typename T26, typename T27, typename T28, typename T29, typename T30,
00700     typename T31>
00701 internal::ValueArray31<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00702     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00703     T29, T30, T31> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7,
00704     T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15,
00705     T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23,
00706     T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31) {
00707     return internal::ValueArray31<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00708         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00709         T26, T27, T28, T29, T30, T31>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10,
00710         v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24,
00711         v25, v26, v27, v28, v29, v30, v31);
00712 }
00713
00714 template <typename T1, typename T2, typename T3, typename T4, typename T5,
```

```

00715     typename T6, typename T7, typename T8, typename T9, typename T10,
00716     typename T11, typename T12, typename T13, typename T14, typename T15,
00717     typename T16, typename T17, typename T18, typename T19, typename T20,
00718     typename T21, typename T22, typename T23, typename T24, typename T25,
00719     typename T26, typename T27, typename T28, typename T29, typename T30,
00720     typename T31, typename T32>
00721 internal::ValueArray32<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00722     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00723     T29, T30, T31, T32> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7,
00724     T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15,
00725     T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23,
00726     T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31,
00727     T32 v32) {
00728     return internal::ValueArray32<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00729         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00730         T26, T27, T28, T29, T30, T31, T32>(v1, v2, v3, v4, v5, v6, v7, v8, v9,
00731         v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23,
00732         v24, v25, v26, v27, v28, v29, v30, v31, v32);
00733 }
00734
00735 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00736     typename T6, typename T7, typename T8, typename T9, typename T10,
00737     typename T11, typename T12, typename T13, typename T14, typename T15,
00738     typename T16, typename T17, typename T18, typename T19, typename T20,
00739     typename T21, typename T22, typename T23, typename T24, typename T25,
00740     typename T26, typename T27, typename T28, typename T29, typename T30,
00741     typename T31, typename T32, typename T33>
00742 internal::ValueArray33<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00743     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00744     T29, T30, T31, T32, T33> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6,
00745     T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15,
00746     T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23,
00747     T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31,
00748     T32 v32, T33 v33) {
00749     return internal::ValueArray33<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00750         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00751         T26, T27, T28, T29, T30, T31, T32, T33>(v1, v2, v3, v4, v5, v6, v7, v8,
00752         v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23,
00753         v24, v25, v26, v27, v28, v29, v30, v31, v32, v33);
00754 }
00755
00756 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00757     typename T6, typename T7, typename T8, typename T9, typename T10,
00758     typename T11, typename T12, typename T13, typename T14, typename T15,
00759     typename T16, typename T17, typename T18, typename T19, typename T20,
00760     typename T21, typename T22, typename T23, typename T24, typename T25,
00761     typename T26, typename T27, typename T28, typename T29, typename T30,
00762     typename T31, typename T32, typename T33, typename T34>
00763 internal::ValueArray34<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00764     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00765     T29, T30, T31, T32, T33, T34> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5,
00766     T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14,
00767     T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22,
00768     T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30,
00769     T31 v31, T32 v32, T33 v33, T34 v34) {
00770     return internal::ValueArray34<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00771         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00772         T26, T27, T28, T29, T30, T31, T32, T33, T34>(v1, v2, v3, v4, v5, v6, v7,
00773         v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22,
00774         v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33, v34);
00775 }
00776
00777 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00778     typename T6, typename T7, typename T8, typename T9, typename T10,
00779     typename T11, typename T12, typename T13, typename T14, typename T15,
00780     typename T16, typename T17, typename T18, typename T19, typename T20,
00781     typename T21, typename T22, typename T23, typename T24, typename T25,
00782     typename T26, typename T27, typename T28, typename T29, typename T30,
00783     typename T31, typename T32, typename T33, typename T34, typename T35>
00784 internal::ValueArray35<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00785     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00786     T29, T30, T31, T32, T33, T34, T35> Values(T1 v1, T2 v2, T3 v3, T4 v4,
00787     T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13,
00788     T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21,
00789     T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29,
00790     T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35) {
00791     return internal::ValueArray35<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00792         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00793         T26, T27, T28, T29, T30, T31, T32, T33, T34, T35>(v1, v2, v3, v4, v5, v6,
00794         v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21,
00795         v22, v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33, v34, v35);
00796 }
00797
00798 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00799     typename T6, typename T7, typename T8, typename T9, typename T10,
00800     typename T11, typename T12, typename T13, typename T14, typename T15,
00801     typename T16, typename T17, typename T18, typename T19, typename T20,

```

```

00802     typename T21, typename T22, typename T23, typename T24, typename T25,
00803     typename T26, typename T27, typename T28, typename T29, typename T30,
00804     typename T31, typename T32, typename T33, typename T34, typename T35,
00805     typename T36>
00806 internal::ValueArray36<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00807     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00808     T29, T30, T31, T32, T33, T34, T35, T36> Values(T1 v1, T2 v2, T3 v3, T4 v4,
00809     T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13,
00810     T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21,
00811     T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29,
00812     T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36) {
00813     return internal::ValueArray36<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00814         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00815         T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36>(v1, v2, v3, v4,
00816         v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19,
00817         v20, v21, v22, v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33,
00818         v34, v35, v36);
00819 }
00820
00821 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00822     typename T6, typename T7, typename T8, typename T9, typename T10,
00823     typename T11, typename T12, typename T13, typename T14, typename T15,
00824     typename T16, typename T17, typename T18, typename T19, typename T20,
00825     typename T21, typename T22, typename T23, typename T24, typename T25,
00826     typename T26, typename T27, typename T28, typename T29, typename T30,
00827     typename T31, typename T32, typename T33, typename T34, typename T35,
00828     typename T36, typename T37>
00829 internal::ValueArray37<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00830     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00831     T29, T30, T31, T32, T33, T34, T35, T36, T37> Values(T1 v1, T2 v2, T3 v3,
00832     T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12,
00833     T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20,
00834     T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28,
00835     T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36,
00836     T37 v37) {
00837     return internal::ValueArray37<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00838         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00839         T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37>(v1, v2, v3,
00840         v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19,
00841         v20, v21, v22, v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33,
00842         v34, v35, v36, v37);
00843 }
00844
00845 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00846     typename T6, typename T7, typename T8, typename T9, typename T10,
00847     typename T11, typename T12, typename T13, typename T14, typename T15,
00848     typename T16, typename T17, typename T18, typename T19, typename T20,
00849     typename T21, typename T22, typename T23, typename T24, typename T25,
00850     typename T26, typename T27, typename T28, typename T29, typename T30,
00851     typename T31, typename T32, typename T33, typename T34, typename T35,
00852     typename T36, typename T37, typename T38>
00853 internal::ValueArray38<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00854     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00855     T29, T30, T31, T32, T33, T34, T35, T36, T37, T38> Values(T1 v1, T2 v2,
00856     T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12,
00857     T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20,
00858     T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28,
00859     T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36,
00860     T37 v37, T38 v38) {
00861     return internal::ValueArray38<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00862         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00863         T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38>(v1, v2,
00864         v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18,
00865         v19, v20, v21, v22, v23, v24, v25, v26, v27, v28, v29, v30, v31, v32,
00866         v33, v34, v35, v36, v37, v38);
00867 }
00868
00869 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00870     typename T6, typename T7, typename T8, typename T9, typename T10,
00871     typename T11, typename T12, typename T13, typename T14, typename T15,
00872     typename T16, typename T17, typename T18, typename T19, typename T20,
00873     typename T21, typename T22, typename T23, typename T24, typename T25,
00874     typename T26, typename T27, typename T28, typename T29, typename T30,
00875     typename T31, typename T32, typename T33, typename T34, typename T35,
00876     typename T36, typename T37, typename T38, typename T39>
00877 internal::ValueArray39<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00878     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00879     T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39> Values(T1 v1, T2 v2,
00880     T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12,
00881     T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20,
00882     T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28,
00883     T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36,
00884     T37 v37, T38 v38, T39 v39) {
00885     return internal::ValueArray39<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00886         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00887         T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39>(v1,
00888         v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17,
```

```

00889     v18, v19, v20, v21, v22, v23, v24, v25, v26, v27, v28, v29, v30, v31,
00890     v32, v33, v34, v35, v36, v37, v38, v39);
00891 }
00892
00893 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00894     typename T6, typename T7, typename T8, typename T9, typename T10,
00895     typename T11, typename T12, typename T13, typename T14, typename T15,
00896     typename T16, typename T17, typename T18, typename T19, typename T20,
00897     typename T21, typename T22, typename T23, typename T24, typename T25,
00898     typename T26, typename T27, typename T28, typename T29, typename T30,
00899     typename T31, typename T32, typename T33, typename T34, typename T35,
00900     typename T36, typename T37, typename T38, typename T39, typename T40>
00901 internal::ValueArray40<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00902     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00903     T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40> Values(T1 v1,
00904     T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11,
00905     T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19,
00906     T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27,
00907     T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35,
00908     T36 v36, T37 v37, T38 v38, T39 v39, T40 v40) {
00909     return internal::ValueArray40<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00910         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00911         T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
00912         T40>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15,
00913         v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26, v27, v28, v29,
00914         v30, v31, v32, v33, v34, v35, v36, v37, v38, v39, v40);
00915 }
00916
00917 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00918     typename T6, typename T7, typename T8, typename T9, typename T10,
00919     typename T11, typename T12, typename T13, typename T14, typename T15,
00920     typename T16, typename T17, typename T18, typename T19, typename T20,
00921     typename T21, typename T22, typename T23, typename T24, typename T25,
00922     typename T26, typename T27, typename T28, typename T29, typename T30,
00923     typename T31, typename T32, typename T33, typename T34, typename T35,
00924     typename T36, typename T37, typename T38, typename T39, typename T40,
00925     typename T41>
00926 internal::ValueArray41<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00927     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00928     T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40,
00929     T41> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00930     T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
00931     T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
00932     T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
00933     T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41) {
00934     return internal::ValueArray41<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00935         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00936         T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
00937         T40, T41>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14,
00938         v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26, v27, v28,
00939         v29, v30, v31, v32, v33, v34, v35, v36, v37, v38, v39, v40, v41);
00940 }
00941
00942 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00943     typename T6, typename T7, typename T8, typename T9, typename T10,
00944     typename T11, typename T12, typename T13, typename T14, typename T15,
00945     typename T16, typename T17, typename T18, typename T19, typename T20,
00946     typename T21, typename T22, typename T23, typename T24, typename T25,
00947     typename T26, typename T27, typename T28, typename T29, typename T30,
00948     typename T31, typename T32, typename T33, typename T34, typename T35,
00949     typename T36, typename T37, typename T38, typename T39, typename T40,
00950     typename T41, typename T42>
00951 internal::ValueArray42<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00952     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00953     T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41,
00954     T42> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00955     T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
00956     T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
00957     T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
00958     T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41,
00959     T42 v42) {
00960     return internal::ValueArray42<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00961         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00962         T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
00963         T40, T41, T42>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13,
00964         v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26, v27,
00965         v28, v29, v30, v31, v32, v33, v34, v35, v36, v37, v38, v39, v40, v41,
00966         v42);
00967 }
00968
00969 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00970     typename T6, typename T7, typename T8, typename T9, typename T10,
00971     typename T11, typename T12, typename T13, typename T14, typename T15,
00972     typename T16, typename T17, typename T18, typename T19, typename T20,
00973     typename T21, typename T22, typename T23, typename T24, typename T25,
00974     typename T26, typename T27, typename T28, typename T29, typename T30,
00975     typename T31, typename T32, typename T33, typename T34, typename T35,
```

```
00976     typename T36, typename T37, typename T38, typename T39, typename T40,
00977     typename T41, typename T42, typename T43>
00978 internal::ValueArray43<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00979     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00980     T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42,
00981     T43> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00982     T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
00983     T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
00984     T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
00985     T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41,
00986     T42 v42, T43 v43) {
00987     return internal::ValueArray43<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00988         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
00989         T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
00990         T40, T41, T42, T43>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12,
00991         v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26,
00992         v27, v28, v29, v30, v31, v32, v33, v34, v35, v36, v37, v38, v39, v40,
00993         v41, v42, v43);
00994 }
00995
00996 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00997     typename T6, typename T7, typename T8, typename T9, typename T10,
00998     typename T11, typename T12, typename T13, typename T14, typename T15,
00999     typename T16, typename T17, typename T18, typename T19, typename T20,
01000     typename T21, typename T22, typename T23, typename T24, typename T25,
01001     typename T26, typename T27, typename T28, typename T29, typename T30,
01002     typename T31, typename T32, typename T33, typename T34, typename T35,
01003     typename T36, typename T37, typename T38, typename T39, typename T40,
01004     typename T41, typename T42, typename T43, typename T44>
01005 internal::ValueArray44<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
01006     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
01007     T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
01008     T44> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
01009     T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
01010     T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
01011     T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
01012     T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41,
01013     T42 v42, T43 v43, T44 v44) {
01014     return internal::ValueArray44<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
01015         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
01016         T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
01017         T40, T41, T42, T43, T44>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11,
01018         v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25,
01019         v26, v27, v28, v29, v30, v31, v32, v33, v34, v35, v36, v37, v38, v39,
01020         v40, v41, v42, v43, v44);
01021 }
01022
01023 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01024     typename T6, typename T7, typename T8, typename T9, typename T10,
01025     typename T11, typename T12, typename T13, typename T14, typename T15,
01026     typename T16, typename T17, typename T18, typename T19, typename T20,
01027     typename T21, typename T22, typename T23, typename T24, typename T25,
01028     typename T26, typename T27, typename T28, typename T29, typename T30,
01029     typename T31, typename T32, typename T33, typename T34, typename T35,
01030     typename T36, typename T37, typename T38, typename T39, typename T40,
01031     typename T41, typename T42, typename T43, typename T44, typename T45>
01032 internal::ValueArray45<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
01033     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
01034     T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
01035     T44, T45> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8,
01036     T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16,
01037     T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24,
01038     T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32,
01039     T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40,
01040     T41 v41, T42 v42, T43 v43, T44 v44, T45 v45) {
01041     return internal::ValueArray45<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
01042         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
01043         T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
01044         T40, T41, T42, T43, T44, T45>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10,
01045         v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24,
01046         v25, v26, v27, v28, v29, v30, v31, v32, v33, v34, v35, v36, v37, v38,
01047         v39, v40, v41, v42, v43, v44, v45);
01048 }
01049
01050 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01051     typename T6, typename T7, typename T8, typename T9, typename T10,
01052     typename T11, typename T12, typename T13, typename T14, typename T15,
01053     typename T16, typename T17, typename T18, typename T19, typename T20,
01054     typename T21, typename T22, typename T23, typename T24, typename T25,
01055     typename T26, typename T27, typename T28, typename T29, typename T30,
01056     typename T31, typename T32, typename T33, typename T34, typename T35,
01057     typename T36, typename T37, typename T38, typename T39, typename T40,
01058     typename T41, typename T42, typename T43, typename T44, typename T45,
01059     typename T46>
01060 internal::ValueArray46<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
01061     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
01062     T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
```

```

01063     T44, T45, T46> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7,
01064     T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15,
01065     T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23,
01066     T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31,
01067     T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39,
01068     T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46) {
01069     return internal::ValueArray46<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
01070     T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
01071     T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
01072     T40, T41, T42, T43, T44, T45, T46>(v1, v2, v3, v4, v5, v6, v7, v8, v9,
01073     v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23,
01074     v24, v25, v26, v27, v28, v29, v30, v31, v32, v33, v34, v35, v36, v37,
01075     v38, v39, v40, v41, v42, v43, v44, v45, v46);
01076 }
01077
01078 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01079     typename T6, typename T7, typename T8, typename T9, typename T10,
01080     typename T11, typename T12, typename T13, typename T14, typename T15,
01081     typename T16, typename T17, typename T18, typename T19, typename T20,
01082     typename T21, typename T22, typename T23, typename T24, typename T25,
01083     typename T26, typename T27, typename T28, typename T29, typename T30,
01084     typename T31, typename T32, typename T33, typename T34, typename T35,
01085     typename T36, typename T37, typename T38, typename T39, typename T40,
01086     typename T41, typename T42, typename T43, typename T44, typename T45,
01087     typename T46, typename T47>
01088     internal::ValueArray47<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
01089     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
01090     T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
01091     T44, T45, T46, T47> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7,
01092     T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15,
01093     T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23,
01094     T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31,
01095     T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39,
01096     T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47) {
01097     return internal::ValueArray47<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
01098     T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
01099     T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
01100     T40, T41, T42, T43, T44, T45, T46, T47>(v1, v2, v3, v4, v5, v6, v7, v8,
01101     v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23,
01102     v24, v25, v26, v27, v28, v29, v30, v31, v32, v33, v34, v35, v36, v37,
01103     v38, v39, v40, v41, v42, v43, v44, v45, v46, v47);
01104 }
01105
01106 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01107     typename T6, typename T7, typename T8, typename T9, typename T10,
01108     typename T11, typename T12, typename T13, typename T14, typename T15,
01109     typename T16, typename T17, typename T18, typename T19, typename T20,
01110     typename T21, typename T22, typename T23, typename T24, typename T25,
01111     typename T26, typename T27, typename T28, typename T29, typename T30,
01112     typename T31, typename T32, typename T33, typename T34, typename T35,
01113     typename T36, typename T37, typename T38, typename T39, typename T40,
01114     typename T41, typename T42, typename T43, typename T44, typename T45,
01115     typename T46, typename T47, typename T48>
01116     internal::ValueArray48<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
01117     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
01118     T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
01119     T44, T45, T46, T47, T48> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6,
01120     T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15,
01121     T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23,
01122     T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31,
01123     T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39,
01124     T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47,
01125     T48 v48) {
01126     return internal::ValueArray48<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
01127     T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
01128     T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
01129     T40, T41, T42, T43, T44, T45, T46, T47, T48>(v1, v2, v3, v4, v5, v6, v7,
01130     v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22,
01131     v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33, v34, v35, v36,
01132     v37, v38, v39, v40, v41, v42, v43, v44, v45, v46, v47, v48);
01133 }
01134
01135 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01136     typename T6, typename T7, typename T8, typename T9, typename T10,
01137     typename T11, typename T12, typename T13, typename T14, typename T15,
01138     typename T16, typename T17, typename T18, typename T19, typename T20,
01139     typename T21, typename T22, typename T23, typename T24, typename T25,
01140     typename T26, typename T27, typename T28, typename T29, typename T30,
01141     typename T31, typename T32, typename T33, typename T34, typename T35,
01142     typename T36, typename T37, typename T38, typename T39, typename T40,
01143     typename T41, typename T42, typename T43, typename T44, typename T45,
01144     typename T46, typename T47, typename T48, typename T49>
01145     internal::ValueArray49<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
01146     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
01147     T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
01148     T44, T45, T46, T47, T48, T49> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5,
01149     T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13, T14 v14,
```

```

01150     T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21, T22 v22,
01151     T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29, T30 v30,
01152     T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37, T38 v38,
01153     T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45, T46 v46,
01154     T47 v47, T48 v48, T49 v49) {
01155     return internal::ValueArray49<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
01156         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
01157         T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
01158         T40, T41, T42, T43, T44, T45, T46, T47, T48, T49>(v1, v2, v3, v4, v5, v6,
01159         v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21,
01160         v22, v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33, v34, v35,
01161         v36, v37, v38, v39, v40, v41, v42, v43, v44, v45, v46, v47, v48, v49);
01162 }
01163
01164 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01165     typename T6, typename T7, typename T8, typename T9, typename T10,
01166     typename T11, typename T12, typename T13, typename T14, typename T15,
01167     typename T16, typename T17, typename T18, typename T19, typename T20,
01168     typename T21, typename T22, typename T23, typename T24, typename T25,
01169     typename T26, typename T27, typename T28, typename T29, typename T30,
01170     typename T31, typename T32, typename T33, typename T34, typename T35,
01171     typename T36, typename T37, typename T38, typename T39, typename T40,
01172     typename T41, typename T42, typename T43, typename T44, typename T45,
01173     typename T46, typename T47, typename T48, typename T49, typename T50>
01174 internal::ValueArray50<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
01175     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
01176     T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
01177     T44, T45, T46, T47, T48, T49, T50> Values(T1 v1, T2 v2, T3 v3, T4 v4,
01178     T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, T11 v11, T12 v12, T13 v13,
01179     T14 v14, T15 v15, T16 v16, T17 v17, T18 v18, T19 v19, T20 v20, T21 v21,
01180     T22 v22, T23 v23, T24 v24, T25 v25, T26 v26, T27 v27, T28 v28, T29 v29,
01181     T30 v30, T31 v31, T32 v32, T33 v33, T34 v34, T35 v35, T36 v36, T37 v37,
01182     T38 v38, T39 v39, T40 v40, T41 v41, T42 v42, T43 v43, T44 v44, T45 v45,
01183     T46 v46, T47 v47, T48 v48, T49 v49, T50 v50) {
01184     return internal::ValueArray50<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
01185         T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
01186         T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
01187         T40, T41, T42, T43, T44, T45, T46, T47, T48, T49, T50>(v1, v2, v3, v4,
01188         v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19,
01189         v20, v21, v22, v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33,
01190         v34, v35, v36, v37, v38, v39, v40, v41, v42, v43, v44, v45, v46, v47,
01191         v48, v49, v50);
01192 }
01193
01194 // Bool() allows generating tests with parameters in a set of (false, true).
01195 //
01196 // Synopsis:
01197 // Bool()
01198 // - returns a generator producing sequences with elements {false, true}.
01199 //
01200 // It is useful when testing code that depends on Boolean flags. Combinations
01201 // of multiple flags can be tested when several Bool()'s are combined using
01202 // Combine() function.
01203 //
01204 // In the following example all tests in the test case FlagDependentTest
01205 // will be instantiated twice with parameters false and true.
01206 //
01207 // class FlagDependentTest : public testing::TestWithParam<bool> {
01208 //     virtual void SetUp() {
01209 //         external_flag = GetParam();
01210 //     }
01211 // }
01212 // INSTANTIATE_TEST_CASE_P(BoolSequence, FlagDependentTest, Bool());
01213 //
01214 inline internal::ParamGenerator<bool> Bool() {
01215     return Values(false, true);
01216 }
01217
01218 # if GTEST_HAS_COMBINE
01219 // Combine() allows the user to combine two or more sequences to produce
01220 // values of a Cartesian product of those sequences' elements.
01221 //
01222 // Synopsis:
01223 // Combine(gen1, gen2, ..., genN)
01224 // - returns a generator producing sequences with elements coming from
01225 //   the Cartesian product of elements from the sequences generated by
01226 //   gen1, gen2, ..., genN. The sequence elements will have a type of
01227 //   tuple<T1, T2, ..., TN> where T1, T2, ..., TN are the types
01228 //   of elements from sequences produced by gen1, gen2, ..., genN.
01229 //
01230 // Combine can have up to 10 arguments. This number is currently limited
01231 // by the maximum number of elements in the tuple implementation used by Google
01232 // Test.
01233 //
01234 // Example:
01235 //
01236 // This will instantiate tests in test case AnimalTest each one with

```

```

01237 // the parameter values tuple("cat", BLACK), tuple("cat", WHITE),
01238 // tuple("dog", BLACK), and tuple("dog", WHITE):
01239 //
01240 // enum Color { BLACK, GRAY, WHITE };
01241 // class AnimalTest
01242 //     : public testing::TestWithParam<tuple<const char*, Color> > {...};
01243 //
01244 // TEST_P(AnimalTest, AnimalLooksNice) {...}
01245 //
01246 // INSTANTIATE_TEST_CASE_P(AnimalVariations, AnimalTest,
01247 //                             Combine(Values("cat", "dog"),
01248 //                                         Values(BLACK, WHITE)));
01249 //
01250 // This will instantiate tests in FlagDependentTest with all variations of two
01251 // Boolean flags:
01252 //
01253 // class FlagDependentTest
01254 //     : public testing::TestWithParam<tuple<bool, bool> > {
01255 //     virtual void SetUp() {
01256 //         // Assigns external_flag_1 and external_flag_2 values from the tuple.
01257 //         tie(external_flag_1, external_flag_2) = GetParam();
01258 //     }
01259 // };
01260 //
01261 // TEST_P(FlagDependentTest, TestFeature1) {
01262 //     // Test your code using external_flag_1 and external_flag_2 here.
01263 // }
01264 // INSTANTIATE_TEST_CASE_P(TwoBoolSequence, FlagDependentTest,
01265 //                           Combine(Bool(), Bool()));
01266 //
01267 template <typename Generator1, typename Generator2>
01268 internal::CartesianProductHolder2<Generator1, Generator2> Combine(
01269     const Generator1& g1, const Generator2& g2) {
01270     return internal::CartesianProductHolder2<Generator1, Generator2>(
01271         g1, g2);
01272 }
01273
01274 template <typename Generator1, typename Generator2, typename Generator3>
01275 internal::CartesianProductHolder3<Generator1, Generator2, Generator3> Combine(
01276     const Generator1& g1, const Generator2& g2, const Generator3& g3) {
01277     return internal::CartesianProductHolder3<Generator1, Generator2, Generator3>(
01278         g1, g2, g3);
01279 }
01280
01281 template <typename Generator1, typename Generator2, typename Generator3,
01282     typename Generator4>
01283 internal::CartesianProductHolder4<Generator1, Generator2, Generator3,
01284     Generator4> Combine(
01285     const Generator1& g1, const Generator2& g2, const Generator3& g3,
01286     const Generator4& g4) {
01287     return internal::CartesianProductHolder4<Generator1, Generator2, Generator3,
01288     Generator4>(
01289         g1, g2, g3, g4);
01290 }
01291
01292 template <typename Generator1, typename Generator2, typename Generator3,
01293     typename Generator4, typename Generator5>
01294 internal::CartesianProductHolder5<Generator1, Generator2, Generator3,
01295     Generator4, Generator5> Combine(
01296     const Generator1& g1, const Generator2& g2, const Generator3& g3,
01297     const Generator4& g4, const Generator5& g5) {
01298     return internal::CartesianProductHolder5<Generator1, Generator2, Generator3,
01299     Generator4, Generator5>(
01300         g1, g2, g3, g4, g5);
01301 }
01302
01303 template <typename Generator1, typename Generator2, typename Generator3,
01304     typename Generator4, typename Generator5, typename Generator6>
01305 internal::CartesianProductHolder6<Generator1, Generator2, Generator3,
01306     Generator4, Generator5, Generator6> Combine(
01307     const Generator1& g1, const Generator2& g2, const Generator3& g3,
01308     const Generator4& g4, const Generator5& g5, const Generator6& g6) {
01309     return internal::CartesianProductHolder6<Generator1, Generator2, Generator3,
01310     Generator4, Generator5, Generator6>(
01311         g1, g2, g3, g4, g5, g6);
01312 }
01313
01314 template <typename Generator1, typename Generator2, typename Generator3,
01315     typename Generator4, typename Generator5, typename Generator6,
01316     typename Generator7>
01317 internal::CartesianProductHolder7<Generator1, Generator2, Generator3,
01318     Generator4, Generator5, Generator6, Generator7> Combine(
01319     const Generator1& g1, const Generator2& g2, const Generator3& g3,
01320     const Generator4& g4, const Generator5& g5, const Generator6& g6,
01321     const Generator7& g7) {
01322     return internal::CartesianProductHolder7<Generator1, Generator2, Generator3,
01323     Generator4, Generator5, Generator6, Generator7>(

```

```

01324     g1, g2, g3, g4, g5, g6, g7);
01325 }
01326
01327 template <typename Generator1, typename Generator2, typename Generator3,
01328     typename Generator4, typename Generator5, typename Generator6,
01329     typename Generator7, typename Generator8>
01330 internal::CartesianProductHolder8<Generator1, Generator2, Generator3,
01331     Generator4, Generator5, Generator6, Generator7, Generator8> Combine(
01332     const Generator1& g1, const Generator2& g2, const Generator3& g3,
01333     const Generator4& g4, const Generator5& g5, const Generator6& g6,
01334     const Generator7& g7, const Generator8& g8) {
01335     return internal::CartesianProductHolder8<Generator1, Generator2, Generator3,
01336     Generator4, Generator5, Generator6, Generator7, Generator8>(
01337         g1, g2, g3, g4, g5, g6, g7, g8);
01338 }
01339
01340 template <typename Generator1, typename Generator2, typename Generator3,
01341     typename Generator4, typename Generator5, typename Generator6,
01342     typename Generator7, typename Generator8, typename Generator9>
01343 internal::CartesianProductHolder9<Generator1, Generator2, Generator3,
01344     Generator4, Generator5, Generator6, Generator7, Generator8,
01345     Generator9> Combine(
01346     const Generator1& g1, const Generator2& g2, const Generator3& g3,
01347     const Generator4& g4, const Generator5& g5, const Generator6& g6,
01348     const Generator7& g7, const Generator8& g8, const Generator9& g9) {
01349     return internal::CartesianProductHolder9<Generator1, Generator2, Generator3,
01350     Generator4, Generator5, Generator6, Generator7, Generator8, Generator9>(
01351         g1, g2, g3, g4, g5, g6, g7, g8, g9);
01352 }
01353
01354 template <typename Generator1, typename Generator2, typename Generator3,
01355     typename Generator4, typename Generator5, typename Generator6,
01356     typename Generator7, typename Generator8, typename Generator9,
01357     typename Generator10>
01358 internal::CartesianProductHolder10<Generator1, Generator2, Generator3,
01359     Generator4, Generator5, Generator6, Generator7, Generator8, Generator9,
01360     Generator10> Combine(
01361     const Generator1& g1, const Generator2& g2, const Generator3& g3,
01362     const Generator4& g4, const Generator5& g5, const Generator6& g6,
01363     const Generator7& g7, const Generator8& g8, const Generator9& g9,
01364     const Generator10& g10) {
01365     return internal::CartesianProductHolder10<Generator1, Generator2, Generator3,
01366     Generator4, Generator5, Generator6, Generator7, Generator8, Generator9,
01367     Generator10>(
01368         g1, g2, g3, g4, g5, g6, g7, g8, g9, g10);
01369 }
01370 # endif // GTEST_HAS_COMBINE
01371
01372 # define TEST_P(test_case_name, test_name) \
01373 class GTEST_TEST_CLASS_NAME_(test_case_name, test_name) \
01374     : public test_case_name { \
01375 public: \
01376     GTEST_TEST_CLASS_NAME_(test_case_name, test_name)() {} \
01377     virtual void TestBody(); \
01378 private: \
01379     static int AddToRegistry() { \
01380         ::testing::UnitTest::GetInstance()->parameterized_test_registry(). \
01381             GetTestCasePatternHolder<test_case_name>(\ 
01382                 #test_case_name, \
01383                 ::testing::internal::CodeLocation(\ 
01384                     __FILE__, __LINE__)->AddTestPattern(\ 
01385                         GTEST_STRINGIFY_(test_case_name), \
01386                         GTEST_STRINGIFY_(test_name), \
01387                         new ::testing::internal::TestMetaFactory< \
01388                             GTEST_TEST_CLASS_NAME_( \
01389                                 test_case_name, test_name)>()); \
01390         return 0; \
01391     } \
01392     static int gtest_registering_dummy_ GTEST_ATTRIBUTE_UNUSED_; \
01393     GTEST_DISALLOW_COPY_AND_ASSIGN_( \
01394         GTEST_TEST_CLASS_NAME_(test_case_name, test_name)); \
01395 }; \
01396 int GTEST_TEST_CLASS_NAME_(test_case_name, \
01397     test_name)::gtest_registering_dummy_ = \
01398     GTEST_TEST_CLASS_NAME_(test_case_name, test_name)::AddToRegistry(); \
01399 void GTEST_TEST_CLASS_NAME_(test_case_name, test_name)::TestBody()
01400
01401 // The optional last argument to INSTANTIATE_TEST_CASE_P allows the user
01402 // to specify a function or functor that generates custom test name suffixes
01403 // based on the test parameters. The function should accept one argument of
01404 // type testing::TestParamInfo<class ParamType>, and return std::string.
01405 //
01406 // testing::PrintToStringParamName is a builtin test suffix generator that
01407 // returns the value of testing::PrintToString(GetParam()).
01408 //
01409 // Note: test names must be non-empty, unique, and may only contain ASCII
01410 // alphanumeric characters or underscore. Because PrintToString adds quotes

```

```

01411 // to std::string and C strings, it won't work for these types.
01412
01413 # define INSTANTIATE_TEST_CASE_P(prefix, test_case_name, generator, ...)
01414 static ::testing::internal::ParamGenerator<test_case_name::ParamType> \
01415     gtest_##prefix##test_case_name##_EvalGenerator_() { return generator; } \
01416 static ::std::string gtest_##prefix##test_case_name##_EvalGenerateName_( \
01417     const ::testing::TestParamInfo<test_case_name::ParamType>& info) { \
01418     return ::testing::internal::GetParamNameGen<test_case_name::ParamType> \
01419         (__VA_ARGS__)(info); \
01420 } \
01421 static int gtest_##prefix##test_case_name##_dummy_ GTEST_ATTRIBUTE_UNUSED_ = \
01422     ::testing::UnitTest::GetInstance()->parameterized_test_registry(). \
01423     GetTestCasePatternHolder<test_case_name>(\ \
01424         #test_case_name, \
01425         ::testing::internal::CodeLocation(\ \
01426             __FILE__, __LINE__))->AddTestCaseInstantiation(\ \
01427             #prefix, \
01428             &gtest_##prefix##test_case_name##_EvalGenerator_, \
01429             &gtest_##prefix##test_case_name##_EvalGenerateName_, \
01430             __FILE__, __LINE__)
01431 }
01432 } // namespace testing
01433
01434 #endif // GTEST_INCLUDE_GTEST_GTEST_PARAM_TEST_H_

```

9.10 Dokumentacja pliku

packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rtdyn.1.8.1.8/build/native/include/gtest/gtest-printers.h

```

#include <iostream>
#include <sstream>
#include <string>
#include <utility>
#include <vector>
#include "gtest/internal/gtest-port.h"
#include "gtest/internal/gtest-internal.h"
#include "gtest/internal/custom/gtest-printers.h"

```

Komponenty

- class [testing::internal2::TypeWithoutFormatter< T, kTypeKind >](#)
- class [testing::internal2::TypeWithoutFormatter< T, kProtobuf >](#)
- class [testing::internal2::TypeWithoutFormatter< T, kConvertibleToInteger >](#)
- class [testing::internal::FormatForComparison< ToPrint, OtherOperand >](#)
- class [testing::internal::FormatForComparison< ToPrint\[N\], OtherOperand >](#)
- struct [testing::internal::WrapPrinterType< type >](#)
- class [testing::internal::UniversalPrinter< T >](#)
- class [testing::internal::UniversalPrinter< T\[N\]>](#)
- class [testing::internal::UniversalPrinter< T & >](#)
- class [testing::internal::UniversalTersePrinter< T >](#)
- class [testing::internal::UniversalTersePrinter< T & >](#)
- class [testing::internal::UniversalTersePrinter< T\[N\]>](#)
- class [testing::internal::UniversalTersePrinter< const char * >](#)
- class [testing::internal::UniversalTersePrinter< char * >](#)
- class [testing::internal::UniversalTersePrinter< wchar_t * >](#)

Przestrzenie nazw

- namespace `testing`
- namespace `testing::internal2`
- namespace `testing_internal`
- namespace `testing::internal`

Definicje

- `#define GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_(CharType)`
- `#define GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(CharType, OtherStringType)`

Definicje typów

- `typedef ::std::vector< ::std::string > testing::internal::Strings`

Wyliczenia

- enum `testing::internal2::TypeKind` { `testing::internal2::kProtobuf` , `testing::internal2::kConvertibleToInteger` , `testing::internal2::kOtherType` }
- enum `testing::internal::DefaultPrinterType` { `testing::internal::kPrintContainer` , `testing::internal::kPrintPointer` , `testing::internal::kPrintFunctionPointer` , `testing::internal::kPrintOther` }

Funkcje

- `GTEST_API_ void testing::internal2::PrintBytesInObjectTo (const unsigned char *obj_bytes, size_t count, ::std::ostream *os)`
- template<typename Char, typename CharTraits, typename T>
`::std::basic_ostream< Char, CharTraits > & testing::internal2::operator<< (::std::basic_ostream< Char, CharTraits > &os, const T &x)`
- template<typename T>
`void testing_internal::DefaultPrintNonContainerTo (const T &value, ::std::ostream *os)`
- `testing::internal::GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_ (char)`
- `testing::internal::GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_ (wchar_t)`
- `testing::internal::GTEST_IMPL_FORMAT_C_STRING_AS_STRING_ (char, ::std::string)`
- template<typename T1, typename T2>
`std::string testing::internal::FormatForComparisonFailureMessage (const T1 &value, const T2 &)`
- template<typename T>
`void testing::internal::UniversalPrint (const T &value, ::std::ostream *os)`
- template<typename C>
`void testing::internal::DefaultPrintTo (WrapPrinterType< kPrintContainer >, const C &container, ::std::ostream *os)`
- template<typename T>
`void testing::internal::DefaultPrintTo (WrapPrinterType< kPrintPointer >, T *p, ::std::ostream *os)`
- template<typename T>
`void testing::internal::DefaultPrintTo (WrapPrinterType< kPrintFunctionPointer >, T *p, ::std::ostream *os)`
- template<typename T>
`void testing::internal::DefaultPrintTo (WrapPrinterType< kPrintOther >, const T &value, ::std::ostream *os)`
- template<typename T>
`void testing::internal::PrintTo (const T &value, ::std::ostream *os)`
- `GTEST_API_ void testing::internal::PrintTo (unsigned char c, ::std::ostream *os)`
- `GTEST_API_ void testing::internal::PrintTo (signed char c, ::std::ostream *os)`

- void `testing::internal::PrintTo` (char c, ::std::ostream *os)
- void `testing::internal::PrintTo` (bool x, ::std::ostream *os)
- `GTEST_API_ void testing::internal::PrintTo` (wchar_t wc, ::std::ostream *os)
- `GTEST_API_ void testing::internal::PrintTo` (const char *s, ::std::ostream *os)
- void `testing::internal::PrintTo` (char *s, ::std::ostream *os)
- void `testing::internal::PrintTo` (const signed char *s, ::std::ostream *os)
- void `testing::internal::PrintTo` (signed char *s, ::std::ostream *os)
- void `testing::internal::PrintTo` (const unsigned char *s, ::std::ostream *os)
- void `testing::internal::PrintTo` (unsigned char *s, ::std::ostream *os)
- `GTEST_API_ void testing::internal::PrintTo` (const wchar_t *s, ::std::ostream *os)
- void `testing::internal::PrintTo` (wchar_t *s, ::std::ostream *os)
- template<typename T>
void `testing::internal::PrintRawArrayTo` (const T a[], size_t count, ::std::ostream *os)
- `GTEST_API_ void testing::internal::PrintStringTo` (const ::std::string &s, ::std::ostream *os)
- void `testing::internal::PrintTo` (const ::std::string &s, ::std::ostream *os)
- template<typename T1, typename T2>
void `testing::internal::PrintTo` (const ::std::pair< T1, T2 > &value, ::std::ostream *os)
- template<typename T>
void `testing::internal::UniversalPrintArray` (const T *begin, size_t len, ::std::ostream *os)
- `GTEST_API_ void testing::internal::UniversalPrintArray` (const char *begin, size_t len, ::std::ostream *os)
- `GTEST_API_ void testing::internal::UniversalPrintArray` (const wchar_t *begin, size_t len, ::std::ostream *os)
- template<typename T>
void `testing::internal::UniversalTersePrint` (const T &value, ::std::ostream *os)
- template<typename T>
::std::string `testing::PrintToString` (const T &value)

Zmienne

- const size_t `testing::internal2::kProtobufOneLinerMaxLength` = 50

9.10.1 Dokumentacja definicji

9.10.1.1 GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_

```
#define GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_(
    CharType)
```

Wartość:

```
template <typename OtherOperand>
class FormatForComparison<CharType*, OtherOperand> {
public:
    static ::std::string Format(CharType* value) {
        return ::testing::PrintToString(static_cast<const void*>(value));
    }
}
```

9.10.1.2 GTEST_IMPL_FORMAT_C_STRING_AS_STRING_

```
#define GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(
    CharType,
    OtherStringType)
```

Wartość:

```
template <>
class FormatForComparison<CharType*, OtherStringType> {
public:
    static ::std::string Format(CharType* value) {
        return ::testing::PrintToString(value);
    }
}
```

9.11 gtest-printers.h

[Idź do dokumentacji tego pliku.](#)

```
00001 // Copyright 2007, Google Inc.
00002 // All rights reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //      * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //      * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029
00030
00031 // Google Test - The Google C++ Testing and Mocking Framework
00032 //
00033 // This file implements a universal value printer that can print a
00034 // value of any type T:
00035 //
00036 void ::testing::internal::UniversalPrinter<T>::Print(value, ostream_ptr);
00037 //
00038 // A user can teach this function how to print a class type T by
00039 // defining either operator«() or PrintTo() in the namespace that
00040 // defines T. More specifically, the FIRST defined function in the
00041 // following list will be used (assuming T is defined in namespace
00042 // foo):
00043 //
00044 // 1. foo::PrintTo(const T&, ostream*)
00045 // 2. operator«(ostream&, const T&) defined in either foo or the
00046 //    global namespace.
00047 //
00048 // However if T is an STL-style container then it is printed element-wise
00049 // unless foo::PrintTo(const T&, ostream*) is defined. Note that
00050 // operator«() is ignored for container types.
00051 //
00052 // If none of the above is defined, it will print the debug string of
00053 // the value if it is a protocol buffer, or print the raw bytes in the
00054 // value otherwise.
00055 //
00056 // To aid debugging: when T is a reference type, the address of the
00057 // value is also printed; when T is a (const) char pointer, both the
00058 // pointer value and the NUL-terminated string it points to are
00059 // printed.
00060 //
00061 // We also provide some convenient wrappers:
00062 //
00063 // Prints a value to a string. For a (const or not) char
00064 // pointer, the NUL-terminated string (but not the pointer) is
00065 // printed.
00066 std::string ::testing::PrintToString(const T& value);
00067 //
00068 // Prints a value tersely: for a reference type, the referenced
00069 // value (but not the address) is printed; for a (const or not) char
00070 // pointer, the NUL-terminated string (but not the pointer) is
00071 // printed.
00072 void ::testing::internal::UniversalTersePrint(const T& value, ostream*);
00073 //
00074 // Prints value using the type inferred by the compiler. The difference
00075 // // from UniversalTersePrint() is that this function prints both the
00076 // // pointer and the NUL-terminated string for a (const or not) char pointer.
00077 void ::testing::internal::UniversalPrint(const T& value, ostream*);
00078 //
00079 // Prints the fields of a tuple tersely to a string vector, one
00080 // // element for each field. Tuple support must be enabled in
00081 // // gtest-port.h.
00082 std::vector<string> UniversalTersePrintTupleFieldsToStrings()
```

```

00083 //         const Tuple& value);
00084 //
00085 // Known limitation:
00086 //
00087 // The print primitives print the elements of an STL-style container
00088 // using the compiler-inferred type of *iter where iter is a
00089 // const_iterator of the container. When const_iterator is an input
00090 // iterator but not a forward iterator, this inferred type may not
00091 // match value_type, and the print output may be incorrect. In
00092 // practice, this is rarely a problem as for most containers
00093 // const_iterator is a forward iterator. We'll fix this if there's an
00094 // actual need for it. Note that this fix cannot rely on value_type
00095 // being defined as many user-defined container types don't have
00096 // value_type.
00097
00098 // GOOGLETEST_CM0001 DO NOT DELETE
00099
00100 #ifndef GTEST_INCLUDE_GTEST_GTEST_PRINTERS_H_
00101 #define GTEST_INCLUDE_GTEST_GTEST_PRINTERS_H_
00102
00103 #include <iostream> // NOLINT
00104 #include <sstream>
00105 #include <string>
00106 #include <utility>
00107 #include <vector>
00108 #include "gtest/internal/gtest-port.h"
00109 #include "gtest/internal/gtest-internal.h"
00110
00111 #if GTEST_HAS_STD_TUPLE_
00112 # include <tuple>
00113 #endif
00114
00115 #if GTEST_HAS_ABSL
00116 #include "absl/strings/string_view.h"
00117 #include "absl/types/optional.h"
00118 #include "absl/types/variant.h"
00119 #endif // GTEST_HAS_ABSL
00120
00121 namespace testing {
00122
00123 // Definitions in the 'internal' and 'internal2' name spaces are
00124 // subject to change without notice. DO NOT USE THEM IN USER CODE!
00125 namespace internal2 {
00126
00127 // Prints the given number of bytes in the given object to the given
00128 // ostream.
00129 GTEST_API_ void PrintBytesInObjectTo(const unsigned char* obj_bytes,
00130                                         size_t count,
00131                                         ::std::ostream* os);
00132
00133 // For selecting which printer to use when a given type has neither «
00134 // nor PrintTo().
00135 enum TypeKind {
00136   kProtobuf,           // a protobuf type
00137   kConvertibleToInteger, // a type implicitly convertible to BiggestInt
00138   // (e.g. a named or unnamed enum type)
00139 #if GTEST_HAS_ABSL
00140   kConvertibleToStringView, // a type implicitly convertible to
00141   // absl::string_view
00142 #endif
00143   kOtherType // anything else
00144 };
00145
00146 // TypeWithoutFormatter<T, kTypeKind>::PrintValue(value, os) is called
00147 // by the universal printer to print a value of type T when neither
00148 // operator« nor PrintTo() is defined for T, where kTypeKind is the
00149 // "kind" of T as defined by enum TypeKind.
00150 template <typename T, TypeKind kTypeKind>
00151 class TypeWithoutFormatter {
00152 public:
00153   // This default version is called when kTypeKind is kOtherType.
00154   static void PrintValue(const T& value, ::std::ostream* os) {
00155     PrintBytesInObjectTo(static_cast<const unsigned char*>(
00156       reinterpret_cast<const void*>(&value)),
00157       sizeof(value), os);
00158   }
00159 };
00160
00161 // We print a protobuf using its ShortDebugString() when the string
00162 // doesn't exceed this many characters; otherwise we print it using
00163 // DebugString() for better readability.
00164 const size_t kProtobufOneLinerMaxLength = 50;
00165
00166 template <typename T>
00167 class TypeWithoutFormatter<T, kProtobuf> {
00168 public:
00169   static void PrintValue(const T& value, ::std::ostream* os) {

```

```

00170     std::string pretty_str = value.ShortDebugString();
00171     if (pretty_str.length() > kProtobufOneLinerMaxLength) {
00172         pretty_str = "\n" + value.DebugString();
00173     }
00174     *os << ("<" + pretty_str + ">");
00175 }
00176 };
00177
00178 template <typename T>
00179 class TypeWithoutFormatter<T, kConvertibleToInteger> {
00180 public:
00181     // Since T has no « operator or PrintTo() but can be implicitly
00182     // converted to BiggestInt, we print it as a BiggestInt.
00183     //
00184     // Most likely T is an enum type (either named or unnamed), in which
00185     // case printing it as an integer is the desired behavior. In case
00186     // T is not an enum, printing it as an integer is the best we can do
00187     // given that it has no user-defined printer.
00188     static void PrintValue(const T& value, ::std::ostream* os) {
00189         const internal::BiggestInt kBBigInt = value;
00190         *os << kBBigInt;
00191     }
00192 };
00193
00194 #if GTEST_HAS_ABSL
00195 template <typename T>
00196 class TypeWithoutFormatter<T, kConvertibleToStringView> {
00197 public:
00198     // Since T has neither operator« nor PrintTo() but can be implicitly
00199     // converted to absl::string_view, we print it as a absl::string_view.
00200     //
00201     // Note: the implementation is further below, as it depends on
00202     // internal::PrintTo symbol which is defined later in the file.
00203     static void PrintValue(const T& value, ::std::ostream* os);
00204 };
00205 #endif
00206
00207 // Prints the given value to the given ostream. If the value is a
00208 // protocol message, its debug string is printed; if it's an enum or
00209 // of a type implicitly convertible to BiggestInt, it's printed as an
00210 // integer; otherwise the bytes in the value are printed. This is
00211 // what UniversalPrinter<T>::Print() does when it knows nothing about
00212 // type T and T has neither « operator nor PrintTo().
00213 //
00214 // A user can override this behavior for a class type Foo by defining
00215 // a « operator in the namespace where Foo is defined.
00216 //
00217 // We put this operator in namespace 'internal2' instead of 'internal'
00218 // to simplify the implementation, as much code in 'internal' needs to
00219 // use « in STL, which would conflict with our own « were it defined
00220 // in 'internal'.
00221 //
00222 // Note that this operator« takes a generic std::basic_ostream<Char,
00223 // CharTraits> type instead of the more restricted std::ostream. If
00224 // we define it to take an std::ostream instead, we'll get an
00225 // "ambiguous overloads" compiler error when trying to print a type
00226 // Foo that supports streaming to std::basic_ostream<Char,
00227 // CharTraits>, as the compiler cannot tell whether
00228 // operator«(std::ostream&, const T&) or
00229 // operator«(std::basic_stream<Char, CharTraits>, const Foo&) is more
00230 // specific.
00231 template <typename Char, typename CharTraits, typename T>
00232 ::std::basic_ostream<Char, CharTraits>& operator<<
00233     ::std::basic_ostream<Char, CharTraits>& os, const T& x) {
00234     TypeWithoutFormatter<T, (internal::IsAProtocolMessage<T>::value
00235         ? kProtobuf
00236         : internal::ImplicitlyConvertible<
00237             const T&, internal::BiggestInt>::value
00238         ? kConvertibleToInteger
00239         :
00240 #if GTEST_HAS_ABSL
00241             internal::ImplicitlyConvertible<
00242                 const T&, absl::string_view>::value
00243             ? kConvertibleToStringView
00244             :
00245 #endif
00246             kOtherType)>::PrintValue(x, &os);
00247     return os;
00248 }
00249
00250 } // namespace internal2
00251 } // namespace testing
00252
00253 // This namespace MUST NOT BE NESTED IN ::testing, or the name look-up
00254 // magic needed for implementing UniversalPrinter won't work.
00255 namespace testing_internal {
00256

```

```

00257 // Used to print a value that is not an STL-style container when the
00258 // user doesn't define PrintTo() for it.
00259 template <typename T>
00260 void DefaultPrintNonContainerTo(const T& value, ::std::ostream* os) {
00261     // With the following statement, during unqualified name lookup,
00262     // testing::internal2::operator« appears as if it was declared in
00263     // the nearest enclosing namespace that contains both
00264     // ::testing_internal and ::testing::internal2, i.e. the global
00265     // namespace. For more details, refer to the C++ Standard section
00266     // 7.3.4-1 [namespace.udir]. This allows us to fall back onto
00267     // testing::internal2::operator« in case T doesn't come with a «
00268     // operator.
00269     //
00270     // We cannot write 'using ::testing::internal2::operator«;', which
00271     // gcc 3.3 fails to compile due to a compiler bug.
00272     using namespace ::testing::internal2; // NOLINT
00273
00274     // Assuming T is defined in namespace foo, in the next statement,
00275     // the compiler will consider all of:
00276     //
00277     // 1. foo::operator« (thanks to Koenig look-up),
00278     // 2. ::operator« (as the current namespace is enclosed in ::),
00279     // 3. testing::internal2::operator« (thanks to the using statement above).
00280     //
00281     // The operator« whose type matches T best will be picked.
00282     //
00283     // We deliberately allow #2 to be a candidate, as sometimes it's
00284     // impossible to define #1 (e.g. when foo is ::std, defining
00285     // anything in it is undefined behavior unless you are a compiler
00286     // vendor.).
00287     *os « value;
00288 }
00289
00290 } // namespace testing_internal
00291
00292 namespace testing {
00293 namespace internal {
00294
00295 // FormatForComparison<ToPoint, OtherOperand>::Format(value) formats a
00296 // value of type ToPrint that is an operand of a comparison assertion
00297 // (e.g. ASSERT_EQ). OtherOperand is the type of the other operand in
00298 // the comparison, and is used to help determine the best way to
00299 // format the value. In particular, when the value is a C string
00300 // (char pointer) and the other operand is an STL string object, we
00301 // want to format the C string as a string, since we know it is
00302 // compared by value with the string object. If the value is a char
00303 // pointer but the other operand is not an STL string object, we don't
00304 // know whether the pointer is supposed to point to a NUL-terminated
00305 // string, and thus want to print it as a pointer to be safe.
00306 //
00307 // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
00308
00309 // The default case.
00310 template <typename ToPrint, typename OtherOperand>
00311 class FormatForComparison {
00312 public:
00313     static ::std::string Format(const ToPrint& value) {
00314         return ::testing::PrintToString(value);
00315     }
00316 };
00317
00318 // Array.
00319 template <typename ToPrint, size_t N, typename OtherOperand>
00320 class FormatForComparison<ToPoint[N], OtherOperand> {
00321 public:
00322     static ::std::string Format(const ToPrint* value) {
00323         return FormatForComparison<const ToPrint*, OtherOperand>::Format(value);
00324     }
00325 };
00326
00327 // By default, print C string as pointers to be safe, as we don't know
00328 // whether they actually point to a NUL-terminated string.
00329
00330 #define GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_(CharType)
00331     template <typename OtherOperand>
00332     class FormatForComparison<CharType*, OtherOperand> {
00333     public:
00334         static ::std::string Format(CharType* value) {
00335             return ::testing::PrintToString(static_cast<const void*>(value));
00336         }
00337     }
00338
00339 GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_(char);
00340 GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_(const char);
00341 GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_(wchar_t);
00342 GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_(const wchar_t);
00343

```

```
00344 #undef GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_
00345
00346 // If a C string is compared with an STL string object, we know it's meant
00347 // to point to a NUL-terminated string, and thus can print it as a string.
00348
00349 #define GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(CharType, OtherStringType) \
00350     template <> \
00351         class FormatForComparison<CharType*, OtherStringType> { \
00352             public: \
00353                 static ::std::string Format(CharType* value) { \
00354                     return ::testing::PrintToString(value); \
00355                 } \
00356         } \
00357
00358 GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(char, ::std::string);
00359 GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(const char, ::std::string);
00360
00361 #if GTEST_HAS_GLOBAL_STRING
00362     GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(char, ::string);
00363     GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(const char, ::string);
00364 #endif
00365
00366 #if GTEST_HAS_GLOBAL_WSTRING
00367     GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(wchar_t, ::wstring);
00368     GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(const wchar_t, ::wstring);
00369 #endif
00370
00371 #if GTEST_HAS_STD_WSTRING
00372     GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(wchar_t, ::std::wstring);
00373     GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(const wchar_t, ::std::wstring);
00374 #endif
00375
00376 #undef GTEST_IMPL_FORMAT_C_STRING_AS_STRING_
00377
00378 // Formats a comparison assertion (e.g. ASSERT_EQ, EXPECT_LT, and etc)
00379 // operand to be used in a failure message. The type (but not value)
00380 // of the other operand may affect the format. This allows us to
00381 // print a char* as a raw pointer when it is compared against another
00382 // char* or void*, and print it as a C string when it is compared
00383 // against an std::string object, for example.
00384 //
00385 // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
00386 template <typename T1, typename T2>
00387 std::string FormatForComparisonFailureMessage(
00388     const T1& value, const T2& /* other_operand */);
00389     return FormatForComparison<T1, T2>::Format(value);
00390 }
00391
00392 // UniversalPrinter<T>::Print(value, ostream_ptr) prints the given
00393 // value to the given ostream. The caller must ensure that
00394 // 'ostream_ptr' is not NULL, or the behavior is undefined.
00395 //
00396 // We define UniversalPrinter as a class template (as opposed to a
00397 // function template), as we need to partially specialize it for
00398 // reference types, which cannot be done with function templates.
00399 template <typename T>
00400 class UniversalPrinter;
00401
00402 template <typename T>
00403 void UniversalPrint(const T& value, ::std::ostream* os);
00404
00405 enum DefaultPrinterType {
00406     kPrintContainer,
00407     kPrintPointer,
00408     kPrintFunctionPointer,
00409     kPrintOther,
00410 };
00411 template <DefaultPrinterType type> struct WrapPrinterType {};
00412
00413 // Used to print an STL-style container when the user doesn't define
00414 // a PrintTo() for it.
00415 template <typename C>
00416 void DefaultPrintTo(WrapPrinterType<kPrintContainer> /* dummy */,
00417     const C& container, ::std::ostream* os) {
00418     const size_t kMaxCount = 32; // The maximum number of elements to print.
00419     *os << '{';
00420     size_t count = 0;
00421     for (typename C::const_iterator it = container.begin();
00422          it != container.end(); ++it, ++count) {
00423         if (count > 0) {
00424             *os << ',';
00425             if (count == kMaxCount) { // Enough has been printed.
00426                 *os << " ...";
00427                 break;
00428             }
00429     }
00430     *os << ' ';
```

```

00431     // We cannot call PrintTo(*it, os) here as PrintTo() doesn't
00432     // handle *it being a native array.
00433     internal::UniversalPrint(*it, os);
00434 }
00435
00436 if (count > 0) {
00437     *os << ' ';
00438 }
00439 *os << '}';
00440 }
00441
00442 // Used to print a pointer that is neither a char pointer nor a member
00443 // pointer, when the user doesn't define PrintTo() for it. (A member
00444 // variable pointer or member function pointer doesn't really point to
00445 // a location in the address space. Their representation is
00446 // implementation-defined. Therefore they will be printed as raw
00447 // bytes.)
00448 template <typename T>
00449 void DefaultPrintTo(WrapPrinterType<kPrintPointer> /* dummy */,
00450                     T* p, ::std::ostream* os) {
00451     if (p == NULL) {
00452         *os << "NULL";
00453     } else {
00454         // T is not a function type. We just call << to print p,
00455         // relying on ADL to pick up user-defined << for their pointer
00456         // types, if any.
00457         *os << p;
00458     }
00459 }
00460 template <typename T>
00461 void DefaultPrintTo(WrapPrinterType<kPrintFunctionPointer> /* dummy */,
00462                     T* p, ::std::ostream* os) {
00463     if (p == NULL) {
00464         *os << "NULL";
00465     } else {
00466         // T is a function type, so *os << p' doesn't do what we want
00467         // (it just prints p as bool). We want to print p as a const
00468         // void*.
00469         *os << reinterpret_cast<const void*>(p);
00470     }
00471 }
00472
00473 // Used to print a non-container, non-pointer value when the user
00474 // doesn't define PrintTo() for it.
00475 template <typename T>
00476 void DefaultPrintTo(WrapPrinterType<kPrintOther> /* dummy */,
00477                      const T& value, ::std::ostream* os) {
00478     ::testing_internal::DefaultPrintNonContainerTo(value, os);
00479 }
00480
00481 // Prints the given value using the << operator if it has one;
00482 // otherwise prints the bytes in it. This is what
00483 // UniversalPrinter<T>::Print() does when PrintTo() is not specialized
00484 // or overloaded for type T.
00485 //
00486 // A user can override this behavior for a class type Foo by defining
00487 // an overload of PrintTo() in the namespace where Foo is defined. We
00488 // give the user this option as sometimes defining a << operator for
00489 // Foo is not desirable (e.g. the coding style may prevent doing it,
00490 // or there is already a << operator but it doesn't do what the user
00491 // wants).
00492 template <typename T>
00493 void PrintTo(const T& value, ::std::ostream* os) {
00494     // DefaultPrintTo() is overloaded. The type of its first argument
00495     // determines which version will be picked.
00496     //
00497     // Note that we check for container types here, prior to we check
00498     // for protocol message types in our operator<<. The rationale is:
00499     //
00500     // For protocol messages, we want to give people a chance to
00501     // override Google Mock's format by defining a PrintTo() or
00502     // operator<<. For STL containers, other formats can be
00503     // incompatible with Google Mock's format for the container
00504     // elements; therefore we check for container types here to ensure
00505     // that our format is used.
00506     //
00507     // Note that MSVC and clang-cl do allow an implicit conversion from
00508     // pointer-to-function to pointer-to-object, but clang-cl warns on it.
00509     // So don't use ImplicitlyConvertible if it can be helped since it will
00510     // cause this warning, and use a separate overload of DefaultPrintTo for
00511     // function pointers so that the `*os << p` in the object pointer overload
00512     // doesn't cause that warning either.
00513     DefaultPrintTo(
00514         WrapPrinterType <
00515             (sizeof(IsContainerTest<T>()) == sizeof(IsContainer)) &&
00516             !IsRecursiveContainer<T>::value
00517             ? kPrintContainer

```

```
00518      : !is_pointer<T>::value
00519          ? kPrintOther
00520 #if GTEST_LANG_CXX11
00521      : std::is_function<typename std::remove_pointer<T>::type>::value
00522 #else
00523      : !internal::ImplicitlyConvertible<T, const void*>::value
00524 #endif
00525          ? kPrintFunctionPointer
00526      : kPrintPointer > (),
00527      value, os);
00528 }
00529
00530 // The following list of PrintTo() overloads tells
00531 // UniversalPrinter<T>::Print() how to print standard types (built-in
00532 // types, strings, plain arrays, and pointers).
00533
00534 // Overloads for various char types.
00535 GTEST_API_ void PrintTo(unsigned char c, ::std::ostream* os);
00536 GTEST_API_ void PrintTo(signed char c, ::std::ostream* os);
00537 inline void PrintTo(char c, ::std::ostream* os) {
00538     // When printing a plain char, we always treat it as unsigned. This
00539     // way, the output won't be affected by whether the compiler thinks
00540     // char is signed or not.
00541     PrintTo(static_cast<unsigned char>(c), os);
00542 }
00543
00544 // Overloads for other simple built-in types.
00545 inline void PrintTo(bool x, ::std::ostream* os) {
00546     *os << (x ? "true" : "false");
00547 }
00548
00549 // Overload for wchar_t type.
00550 // Prints a wchar_t as a symbol if it is printable or as its internal
00551 // code otherwise and also as its decimal code (except for L'\0').
00552 // The L'\0' char is printed as "L'\\0'. The decimal code is printed
00553 // as signed integer when wchar_t is implemented by the compiler
00554 // as a signed type and is printed as an unsigned integer when wchar_t
00555 // is implemented as an unsigned type.
00556 GTEST_API_ void PrintTo(wchar_t wc, ::std::ostream* os);
00557
00558 // Overloads for C strings.
00559 GTEST_API_ void PrintTo(const char* s, ::std::ostream* os);
00560 inline void PrintTo(char* s, ::std::ostream* os) {
00561     PrintTo(ImplicitCast<const char*>(s), os);
00562 }
00563
00564 // signed/unsigned char is often used for representing binary data, so
00565 // we print pointers to it as void* to be safe.
00566 inline void PrintTo(const signed char* s, ::std::ostream* os) {
00567     PrintTo(ImplicitCast<const void*>(s), os);
00568 }
00569 inline void PrintTo(signed char* s, ::std::ostream* os) {
00570     PrintTo(ImplicitCast<const void*>(s), os);
00571 }
00572 inline void PrintTo(const unsigned char* s, ::std::ostream* os) {
00573     PrintTo(ImplicitCast<const void*>(s), os);
00574 }
00575 inline void PrintTo(unsigned char* s, ::std::ostream* os) {
00576     PrintTo(ImplicitCast<const void*>(s), os);
00577 }
00578
00579 // MSVC can be configured to define wchar_t as a typedef of unsigned
00580 // short. It defines _NATIVE_WCHAR_T_DEFINED when wchar_t is a native
00581 // type. When wchar_t is a typedef, defining an overload for const
00582 // wchar_t* would cause unsigned short* be printed as a wide string,
00583 // possibly causing invalid memory accesses.
00584 #if !defined(_MSC_VER) || defined(_NATIVE_WCHAR_T_DEFINED)
00585 // Overloads for wide C strings
00586 GTEST_API_ void PrintTo(const wchar_t* s, ::std::ostream* os);
00587 inline void PrintTo(wchar_t* s, ::std::ostream* os) {
00588     PrintTo(ImplicitCast<const wchar_t*>(s), os);
00589 }
00590 #endif
00591
00592 // Overload for C arrays. Multi-dimensional arrays are printed
00593 // properly.
00594
00595 // Prints the given number of elements in an array, without printing
00596 // the curly braces.
00597 template <typename T>
00598 void PrintRawArrayTo(const T a[], size_t count, ::std::ostream* os) {
00599     UniversalPrint(a[0], os);
00600     for (size_t i = 1; i != count; i++) {
00601         *os << ", ";
00602         UniversalPrint(a[i], os);
00603     }
00604 }
```

```
00605 // Overloads for ::string and ::std::string.
00606 #if GTEST_HAS_GLOBAL_STRING
00607 GTEST_API_ void PrintStringTo(const ::string& s, ::std::ostream* os);
00608 inline void PrintTo(const ::string& s, ::std::ostream* os) {
00609   PrintStringTo(s, os);
00610 }
00612 #endif // GTEST_HAS_GLOBAL_STRING
00613
00614 GTEST_API_ void PrintStringTo(const ::std::string& s, ::std::ostream* os);
00615 inline void PrintTo(const ::std::string& s, ::std::ostream* os) {
00616   PrintStringTo(s, os);
00617 }
00618
00619 // Overloads for ::wstring and ::std::wstring.
00620 #if GTEST_HAS_GLOBAL_WSTRING
00621 GTEST_API_ void PrintWideStringTo(const ::wstring& s, ::std::ostream* os);
00622 inline void PrintTo(const ::wstring& s, ::std::ostream* os) {
00623   PrintWideStringTo(s, os);
00624 }
00625 #endif // GTEST_HAS_GLOBAL_WSTRING
00626
00627 #if GTEST_HAS_STD_WSTRING
00628 GTEST_API_ void PrintWideStringTo(const ::std::wstring& s, ::std::ostream* os);
00629 inline void PrintTo(const ::std::wstring& s, ::std::ostream* os) {
00630   PrintWideStringTo(s, os);
00631 }
00632 #endif // GTEST_HAS_STD_WSTRING
00633
00634 #if GTEST_HAS_ABSL
00635 // Overload for absl::string_view.
00636 inline void PrintTo(absl::string_view sp, ::std::ostream* os) {
00637   PrintTo(::std::string(sp), os);
00638 }
00639 #endif // GTEST_HAS_ABSL
00640
00641 #if GTEST_LANG_CXX11
00642 inline void PrintTo(std::nullptr_t, ::std::ostream* os) { *os << "(nullptr)"; }
00643 #endif // GTEST_LANG_CXX11
00644
00645 #if GTEST_HAS_TR1_TUPLE || GTEST_HAS_STD_TUPLE_
00646 // Helper function for printing a tuple.  T must be instantiated with
00647 // a tuple type.
00648 template <typename T>
00649 void PrintTupleTo(const T& t, ::std::ostream* os);
00650 #endif // GTEST_HAS_TR1_TUPLE || GTEST_HAS_STD_TUPLE_
00651
00652 #if GTEST_HAS_TR1_TUPLE
00653 // Overload for ::std::tr1::tuple.  Needed for printing function arguments,
00654 // which are packed as tuples.
00655
00656 // Overloaded PrintTo() for tuples of various arities.  We support
00657 // tuples of up-to 10 fields.  The following implementation works
00658 // regardless of whether tr1::tuple is implemented using the
00659 // non-standard variadic template feature or not.
00660
00661 inline void PrintTo(const ::std::tr1::tuple<>& t, ::std::ostream* os) {
00662   PrintTupleTo(t, os);
00663 }
00664
00665 template <typename T1>
00666 void PrintTo(const ::std::tr1::tuple<T1>& t, ::std::ostream* os) {
00667   PrintTupleTo(t, os);
00668 }
00669
00670 template <typename T1, typename T2>
00671 void PrintTo(const ::std::tr1::tuple<T1, T2>& t, ::std::ostream* os) {
00672   PrintTupleTo(t, os);
00673 }
00674
00675 template <typename T1, typename T2, typename T3>
00676 void PrintTo(const ::std::tr1::tuple<T1, T2, T3>& t, ::std::ostream* os) {
00677   PrintTupleTo(t, os);
00678 }
00679
00680 template <typename T1, typename T2, typename T3, typename T4>
00681 void PrintTo(const ::std::tr1::tuple<T1, T2, T3, T4>& t, ::std::ostream* os) {
00682   PrintTupleTo(t, os);
00683 }
00684
00685 template <typename T1, typename T2, typename T3, typename T4, typename T5>
00686 void PrintTo(const ::std::tr1::tuple<T1, T2, T3, T4, T5>& t,
00687               ::std::ostream* os) {
00688   PrintTupleTo(t, os);
00689 }
00690
00691 template <typename T1, typename T2, typename T3, typename T4, typename T5>
```

```
00692         typename T6>
00693 void PrintTo(const ::std::tr1::tuple<T1, T2, T3, T4, T5, T6>& t,
00694               ::std::ostream* os) {
00695     PrintTupleTo(t, os);
00696 }
00697
00698 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00699             typename T6, typename T7>
00700 void PrintTo(const ::std::tr1::tuple<T1, T2, T3, T4, T5, T6, T7>& t,
00701               ::std::ostream* os) {
00702     PrintTupleTo(t, os);
00703 }
00704
00705 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00706             typename T6, typename T7, typename T8>
00707 void PrintTo(const ::std::tr1::tuple<T1, T2, T3, T4, T5, T6, T7, T8>& t,
00708               ::std::ostream* os) {
00709     PrintTupleTo(t, os);
00710 }
00711
00712 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00713             typename T6, typename T7, typename T8, typename T9>
00714 void PrintTo(const ::std::tr1::tuple<T1, T2, T3, T4, T5, T6, T7, T8, T9>& t,
00715               ::std::ostream* os) {
00716     PrintTupleTo(t, os);
00717 }
00718
00719 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00720             typename T6, typename T7, typename T8, typename T9, typename T10>
00721 void PrintTo(
00722   const ::std::tr1::tuple<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10>& t,
00723   ::std::ostream* os) {
00724     PrintTupleTo(t, os);
00725 }
00726 #endif // GTEST_HAS_TR1_TUPLE
00727
00728 #if GTEST_HAS_STD_TUPLE_
00729 template <typename... Types>
00730 void PrintTo(const ::std::tuple<Types...>& t, ::std::ostream* os) {
00731     PrintTupleTo(t, os);
00732 }
00733 #endif // GTEST_HAS_STD_TUPLE_
00734
00735 // Overload for std::pair.
00736 template <typename T1, typename T2>
00737 void PrintTo(const ::std::pair<T1, T2>& value, ::std::ostream* os) {
00738     *os << '(';
00739     // We cannot use UniversalPrint(value.first, os) here, as T1 may be
00740     // a reference type. The same for printing value.second.
00741     UniversalPrinter<T1>::Print(value.first, os);
00742     *os << ", ";
00743     UniversalPrinter<T2>::Print(value.second, os);
00744     *os << ')';
00745 }
00746
00747 // Implements printing a non-reference type T by letting the compiler
00748 // pick the right overload of PrintTo() for T.
00749 template <typename T>
00750 class UniversalPrinter {
00751 public:
00752     // MSVC warns about adding const to a function type, so we want to
00753     // disable the warning.
00754     GTEST_DISABLE_MSC_WARNINGS_PUSH_(4180)
00755
00756     // Note: we deliberately don't call this PrintTo(), as that name
00757     // conflicts with ::testing::internal::PrintTo in the body of the
00758     // function.
00759     static void Print(const T& value, ::std::ostream* os) {
00760         // By default, ::testing::internal::PrintTo() is used for printing
00761         // the value.
00762         //
00763         // Thanks to Koenig look-up, if T is a class and has its own
00764         // PrintTo() function defined in its namespace, that function will
00765         // be visible here. Since it is more specific than the generic ones
00766         // in ::testing::internal, it will be picked by the compiler in the
00767         // following statement - exactly what we want.
00768         PrintTo(value, os);
00769     }
00770
00771     GTEST_DISABLE_MSC_WARNINGS_POP_()
00772 };
00773
00774 #if GTEST_HAS_ABSL
00775
00776 // Printer for absl::optional
00777
00778 template <typename T>
```

```

00779 class UniversalPrinter<::absl::optional<T> {
00780 public:
00781     static void Print(const ::absl::optional<T>& value, ::std::ostream* os) {
00782         *os << '(';
00783         if (!value) {
00784             *os << "nullopt";
00785         } else {
00786             UniversalPrint(*value, os);
00787         }
00788         *os << ')';
00789     }
00790 };
00791
00792 // Printer for absl::variant
00793
00794 template <typename... T>
00795 class UniversalPrinter<::absl::variant<T...> {
00796 public:
00797     static void Print(const ::absl::variant<T...>& value, ::std::ostream* os) {
00798         *os << '(';
00799         absl::visit(Visitor{os}, value);
00800         *os << ')';
00801     }
00802
00803 private:
00804     struct Visitor {
00805         template <typename U>
00806         void operator()(const U& u) const {
00807             *os << "/" << GetTypeName<U>() << "' with value ";
00808             UniversalPrint(u, os);
00809         }
00810         ::std::ostream* os;
00811     };
00812 };
00813
00814 #endif // GTEST_HAS_ABSL
00815
00816 // UniversalPrintArray(begin, len, os) prints an array of 'len'
00817 // elements, starting at address 'begin'.
00818 template <typename T>
00819 void UniversalPrintArray(const T* begin, size_t len, ::std::ostream* os) {
00820     if (len == 0) {
00821         *os << "{}";
00822     } else {
00823         *os << "{ ";
00824         const size_t kThreshold = 18;
00825         const size_t kChunkSize = 8;
00826         // If the array has more than kThreshold elements, we'll have to
00827         // omit some details by printing only the first and the last
00828         // kChunkSize elements.
00829         // FIXME: let the user control the threshold using a flag.
00830         if (len <= kThreshold) {
00831             PrintRawArrayTo(begin, len, os);
00832         } else {
00833             PrintRawArrayTo(begin, kChunkSize, os);
00834             *os << ", ..., ";
00835             PrintRawArrayTo(begin + len - kChunkSize, kChunkSize, os);
00836         }
00837         *os << " }";
00838     }
00839 }
00840 // This overload prints a (const) char array compactly.
00841 GTEST_API_ void UniversalPrintArray(
00842     const char* begin, size_t len, ::std::ostream* os);
00843
00844 // This overload prints a (const) wchar_t array compactly.
00845 GTEST_API_ void UniversalPrintArray(
00846     const wchar_t* begin, size_t len, ::std::ostream* os);
00847
00848 // Implements printing an array type T[N].
00849 template <typename T, size_t N>
00850 class UniversalPrinter<T[N]> {
00851 public:
00852     // Prints the given array, omitting some elements when there are too
00853     // many.
00854     static void Print(const T (&a)[N], ::std::ostream* os) {
00855         UniversalPrintArray(a, N, os);
00856     }
00857 };
00858
00859 // Implements printing a reference type T&.
00860 template <typename T>
00861 class UniversalPrinter<T&> {
00862 public:
00863     // MSVC warns about adding const to a function type, so we want to
00864     // disable the warning.
00865     GTEST_DISABLE_MSC_WARNINGS_PUSH_(4180)

```

```
00866
00867     static void Print(const T& value, ::std::ostream* os) {
00868         // Prints the address of the value. We use reinterpret_cast here
00869         // as static_cast doesn't compile when T is a function type.
00870         *os << "@" << reinterpret_cast<const void*>(&value) << " ";
00871
00872         // Then prints the value itself.
00873         UniversalPrint(value, os);
00874     }
00875
00876     GTEST_DISABLE_MSC_WARNINGS_POP_()
00877 };
00878
00879 // Prints a value tersely: for a reference type, the referenced value
00880 // (but not the address) is printed; for a (const) char pointer, the
00881 // NUL-terminated string (but not the pointer) is printed.
00882
00883 template <typename T>
00884 class UniversalTersePrinter {
00885     public:
00886     static void Print(const T& value, ::std::ostream* os) {
00887         UniversalPrint(value, os);
00888     }
00889 };
00890 template <typename T>
00891 class UniversalTersePrinter<T&gt; {
00892     public:
00893     static void Print(const T& value, ::std::ostream* os) {
00894         UniversalPrint(value, os);
00895     }
00896 };
00897 template <typename T, size_t N>
00898 class UniversalTersePrinter<T[N]> {
00899     public:
00900     static void Print(const T (&value)[N], ::std::ostream* os) {
00901         UniversalPrinter<T[N]>::Print(value, os);
00902     }
00903 };
00904 template <>
00905 class UniversalTersePrinter<const char*> {
00906     public:
00907     static void Print(const char* str, ::std::ostream* os) {
00908         if (str == NULL) {
00909             *os << "NULL";
00910         } else {
00911             UniversalPrint(std::string(str), os);
00912         }
00913     }
00914 };
00915 template <>
00916 class UniversalTersePrinter<char*> {
00917     public:
00918     static void Print(char* str, ::std::ostream* os) {
00919         UniversalTersePrinter<const char*>::Print(str, os);
00920     }
00921 };
00922
00923 #if GTEST_HAS_STD_WSTRING
00924 template <>
00925 class UniversalTersePrinter<const wchar_t*> {
00926     public:
00927     static void Print(const wchar_t* str, ::std::ostream* os) {
00928         if (str == NULL) {
00929             *os << "NULL";
00930         } else {
00931             UniversalPrint(::std::wstring(str), os);
00932         }
00933     }
00934 };
00935 #endif
00936
00937 template <>
00938 class UniversalTersePrinter<wchar_t*> {
00939     public:
00940     static void Print(wchar_t* str, ::std::ostream* os) {
00941         UniversalTersePrinter<const wchar_t*>::Print(str, os);
00942     }
00943 };
00944
00945 template <typename T>
00946 void UniversalTersePrint(const T& value, ::std::ostream* os) {
00947     UniversalTersePrinter<T>::Print(value, os);
00948 }
00949
00950 // Prints a value using the type inferred by the compiler. The
00951 // difference between this and UniversalTersePrint() is that for a
00952 // (const) char pointer, this prints both the pointer and the
```

```

00953 // NUL-terminated string.
00954 template <typename T>
00955 void UniversalPrint(const T& value, ::std::ostream* os) {
00956     // A workaround for the bug in VC++ 7.1 that prevents us from instantiating
00957     // UniversalPrinter with T directly.
00958     typedef T T1;
00959     UniversalPrinter<T1>::Print(value, os);
00960 }
00961
00962 typedef ::std::vector< ::std::string> Strings;
00963
00964 // TuplePolicy<TupleT> must provide:
00965 // - tuple_size
00966 //   size of tuple TupleT.
00967 // - get<size_t I>(const TupleT& t)
00968 //   static function extracting element I of tuple TupleT.
00969 // - tuple_element<size_t I>::type
00970 //   type of element I of tuple TupleT.
00971 template <typename TupleT>
00972 struct TuplePolicy;
00973
00974 #if GTEST_HAS_TR1_TUPLE
00975 template <typename TupleT>
00976 struct TuplePolicy {
00977     typedef TupleT Tuple;
00978     static const size_t tuple_size = ::std::tr1::tuple_size<Tuple>::value;
00979
00980     template <size_t I>
00981     struct tuple_element : ::std::tr1::tuple_element<static_cast<int>(I), Tuple> {
00982     };
00983
00984     template <size_t I>
00985     static typename AddReference<const typename ::std::tr1::tuple_element<
00986         static_cast<int>(I), Tuple>::type>::type
00987     get(const Tuple& tuple) {
00988         return ::std::tr1::get<I>(tuple);
00989     }
00990 };
00991 template <typename TupleT>
00992 const size_t TuplePolicy<TupleT>::tuple_size;
00993 #endif // GTEST_HAS_TR1_TUPLE
00994
00995 #if GTEST_HAS_STD_TUPLE_
00996 template <typename... Types>
00997 struct TuplePolicy< ::std::tuple<Types...> > {
00998     typedef ::std::tuple<Types...> Tuple;
00999     static const size_t tuple_size = ::std::tuple_size<Tuple>::value;
01000
01001     template <size_t I>
01002     struct tuple_element : ::std::tuple_element<I, Tuple> {};
01003
01004     template <size_t I>
01005     static const typename ::std::tuple_element<I, Tuple>::type& get(
01006         const Tuple& tuple) {
01007         return ::std::get<I>(tuple);
01008     }
01009 };
01010 template <typename... Types>
01011 const size_t TuplePolicy< ::std::tuple<Types...> >::tuple_size;
01012 #endif // GTEST_HAS_STD_TUPLE_
01013
01014 #if GTEST_HAS_TR1_TUPLE || GTEST_HAS_STD_TUPLE_
01015 // This helper template allows PrintTo() for tuples and
01016 // UniversalTersePrintTupleFieldsToStrings() to be defined by
01017 // induction on the number of tuple fields. The idea is that
01018 // TuplePrefixPrinter<N>::PrintPrefixTo(t, os) prints the first N
01019 // fields in tuple t, and can be defined in terms of
01020 // TuplePrefixPrinter<N - 1>.
01021 //
01022 // The inductive case.
01023 template <size_t N>
01024 struct TuplePrefixPrinter {
01025     // Prints the first N fields of a tuple.
01026     template <typename Tuple>
01027     static void PrintPrefixTo(const Tuple& t, ::std::ostream* os) {
01028         TuplePrefixPrinter<N - 1>::PrintPrefixTo(t, os);
01029         GTEST_INTENTIONAL_CONST_COND_PUSH_()
01030         if (N > 1) {
01031             GTEST_INTENTIONAL_CONST_COND_POP_()
01032             *os << ", ";
01033         }
01034         UniversalPrinter<
01035             typename TuplePolicy<Tuple>::template tuple_element<N - 1>::type>
01036             ::Print(TuplePolicy<Tuple>::template get<N - 1>(t), os);
01037     }
01038
01039     // Tersely prints the first N fields of a tuple to a string vector,

```

```
01040 // one element for each field.
01041 template <typename Tuple>
01042 static void TersePrintPrefixToStrings(const Tuple& t, Strings* strings) {
01043     TuplePrefixPrinter<N - 1>::TersePrintPrefixToStrings(t, strings);
01044     ::std::stringstream ss;
01045     UniversalTersePrint(TuplePolicy<Tuple>::template get<N - 1>(t), &ss);
01046     strings->push_back(ss.str());
01047 }
01048 };
01049
01050 // Base case.
01051 template <>
01052 struct TuplePrefixPrinter<0> {
01053     template <typename Tuple>
01054     static void PrintPrefixTo(const Tuple&, ::std::ostream*) {}
01055
01056     template <typename Tuple>
01057     static void TersePrintPrefixToStrings(const Tuple&, Strings*) {}
01058 };
01059
01060 // Helper function for printing a tuple.
01061 // Tuple must be either std::tr1::tuple or std::tuple type.
01062 template <typename Tuple>
01063 void PrintTupleTo(const Tuple& t, ::std::ostream* os) {
01064     *os << "(";
01065     TuplePrefixPrinter<TuplePolicy<Tuple>::tuple_size>::PrintPrefixTo(t, os);
01066     *os << ")";
01067 }
01068
01069 // Prints the fields of a tuple tersely to a string vector, one
01070 // element for each field. See the comment before
01071 // UniversalTersePrint() for how we define "tersely".
01072 template <typename Tuple>
01073 Strings UniversalTersePrintTupleFieldsToStrings(const Tuple& value) {
01074     Strings result;
01075     TuplePrefixPrinter<TuplePolicy<Tuple>::tuple_size>::
01076         TersePrintPrefixToStrings(value, &result);
01077     return result;
01078 }
01079 #endif // GTEST_HAS_TR1_TUPLE || GTEST_HAS_STD_TUPLE_
01080
01081 } // namespace internal
01082
01083 #if GTEST_HAS_ABSL
01084 namespace internal12 {
01085     template <typename T>
01086     void TypeWithoutFormatter<T, kConvertibleToStringView>::PrintValue(
01087         const T& value, ::std::ostream* os) {
01088         internal12::PrintTo(absl::string_view(value), os);
01089     }
01090 } // namespace internal12
01091 #endif
01092
01093 template <typename T>
01094 ::std::string PrintToString(const T& value) {
01095     ::std::stringstream ss;
01096     internal12::UniversalTersePrinter<T>::Print(value, &ss);
01097     return ss.str();
01098 }
01099
01100 } // namespace testing
01101
01102 // Include any custom printer added by the local installation.
01103 // We must include this header at the end to make sure it can use the
01104 // declarations from this file.
01105 #include "gtest/internal/custom/gtest-printers.h"
01106
01107 #endif // GTEST_INCLUDE_GTEST_GTEST_PRINTERS_H_
```

9.12 Dokumentacja pliku

packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/custom/gtest-printers.h

9.13 gtest-printers.h

[Idź do dokumentacji tego pliku.](#)

```

00001 // Copyright 2015, Google Inc.
00002 // All rights reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //      * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //      * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 // This file provides an injection point for custom printers in a local
00031 // installation of gTest.
00032 // It will be included from gtest-printers.h and the overrides in this file
00033 // will be visible to everyone.
00034 //
00035 // Injection point for custom user configurations. See README for details
00036 //
00037 // ** Custom implementation starts here **
00038
00039 #ifndef GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_PRINTERS_H_
00040 #define GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_PRINTERS_H_
00041
00042 #endif // GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_PRINTERS_H_

```

9.14 Dokumentacja pliku

packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-spi.h

```
#include "gtest/gtest.h"
```

Definicje

- `#define EXPECT_FATAL_FAILURE(statement, substr)`
- `#define EXPECT_FATAL_FAILURE_ON_ALL_THREADS(statement, substr)`
- `#define EXPECT_NONFATAL_FAILURE(statement, substr)`
- `#define EXPECT_NONFATAL_FAILURE_ON_ALL_THREADS(statement, substr)`

Funkcje

- `GTEST_DISABLE_MSC_WARNINGS_PUSH_(4251)` namespace `testing`

9.14.1 Dokumentacja definicji

9.14.1.1 EXPECT_FATAL_FAILURE

```
#define EXPECT_FATAL_FAILURE(
    statement,
    substr)
```

Wartość:

```
do { \
    class GTestExpectFatalFailureHelper {
public:
    static void Execute() { statement; } \
}; \
::testing::TestPartResultArray gtest_failures; \
::testing::internal::SingleFailureChecker gtest_checker( \
    &gtest_failures, ::testing::TestPartResult::kFatalFailure, (substr)); \
{ \
    ::testing::ScopedFakeTestPartResultReporter gtest_reporter( \
        ::testing::ScopedFakeTestPartResultReporter:: \
        INTERCEPT_ONLY_CURRENT_THREAD, &gtest_failures); \
    GTestExpectFatalFailureHelper::Execute(); \
} \
} while (::testing::internal::AlwaysFalse())
```

9.14.1.2 EXPECT_FATAL_FAILURE_ON_ALL_THREADS

```
#define EXPECT_FATAL_FAILURE_ON_ALL_THREADS(
    statement,
    substr)
```

Wartość:

```
do { \
    class GTestExpectFatalFailureHelper {
public:
    static void Execute() { statement; } \
}; \
::testing::TestPartResultArray gtest_failures; \
::testing::internal::SingleFailureChecker gtest_checker( \
    &gtest_failures, ::testing::TestPartResult::kFatalFailure, (substr)); \
{ \
    ::testing::ScopedFakeTestPartResultReporter gtest_reporter( \
        ::testing::ScopedFakeTestPartResultReporter:: \
        INTERCEPT_ALL_THREADS, &gtest_failures); \
    GTestExpectFatalFailureHelper::Execute(); \
} \
} while (::testing::internal::AlwaysFalse())
```

9.14.1.3 EXPECT_NONFATAL_FAILURE

```
#define EXPECT_NONFATAL_FAILURE(
    statement,
    substr)
```

Wartość:

```
do { \
    ::testing::TestPartResultArray gtest_failures; \
    ::testing::internal::SingleFailureChecker gtest_checker( \
        &gtest_failures, ::testing::TestPartResult::kNonFatalFailure, \
        (substr)); \
{ \
    ::testing::ScopedFakeTestPartResultReporter gtest_reporter( \
        ::testing::ScopedFakeTestPartResultReporter:: \
        INTERCEPT_ONLY_CURRENT_THREAD, &gtest_failures); \
    if (::testing::internal::AlwaysTrue()) { statement; } \
} \
} while (::testing::internal::AlwaysFalse())
```

9.14.1.4 EXPECT_NONFATAL_FAILURE_ON_ALL_THREADS

```
#define EXPECT_NONFATAL_FAILURE_ON_ALL_THREADS( \
    statement, \
    substr)
```

Wartość:

```
do { \
    ::testing::TestPartResultArray gtest_failures; \
    ::testing::internal::SingleFailureChecker gtest_checker( \
        &gtest_failures, ::testing::TestPartResult::kNonFatalFailure, \
        (substr)); \
    { \
        ::testing::ScopedFakeTestPartResultReporter gtest_reporter( \
            ::testing::ScopedFakeTestPartResultReporter::INTERCEPT_ALL_THREADS, \
            &gtest_failures); \
        if (::testing::internal::AlwaysTrue()) { statement; } \
    } \
} while (::testing::internal::AlwaysFalse())
```

9.14.2 Dokumentacja funkcji

9.14.2.1 GTEST_DISABLE_MSC_WARNINGS_PUSH_()

```
GTEST_DISABLE_MSC_WARNINGS_PUSH_ ( \
    4251 )
```

9.15 gtest-spi.h

[Idź do dokumentacji tego pliku.](#)

```
00001 // Copyright 2007, Google Inc. \
00002 // All rights reserved. \
00003 // \
00004 // Redistribution and use in source and binary forms, with or without \
00005 // modification, are permitted provided that the following conditions are \
00006 // met: \
00007 // \
00008 //     * Redistributions of source code must retain the above copyright \
00009 // notice, this list of conditions and the following disclaimer. \
00010 //     * Redistributions in binary form must reproduce the above \
00011 // copyright notice, this list of conditions and the following disclaimer \
00012 // in the documentation and/or other materials provided with the \
00013 // distribution. \
00014 //     * Neither the name of Google Inc. nor the names of its \
00015 // contributors may be used to endorse or promote products derived from \
00016 // this software without specific prior written permission. \
00017 // \
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS \
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT \
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR \
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT \
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, \
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT \
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, \
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY \
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT \
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE \
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. \
00029 // \
00030 // \
00031 // Utilities for testing Google Test itself and code that uses Google Test \
00032 // (e.g. frameworks built on top of Google Test). \
00033 // \
00034 // GOOGLETST_CM0004 DO NOT DELETE \
00035 // \
00036 #ifndef GTEST_INCLUDE_GTEST_GTEST_SPI_H_ \
00037 #define GTEST_INCLUDE_GTEST_GTEST_SPI_H_ \
00038 // \
00039 #include "gtest/gtest.h"
```

```
00040
00041 GTEST_DISABLE_MSC_WARNINGS_PUSH_(4251 \
00042 /* class A needs to have dll-interface to be used by clients of class B */)
00043
00044 namespace testing {
00045
00046 // This helper class can be used to mock out Google Test failure reporting
00047 // so that we can test Google Test or code that builds on Google Test.
00048 //
00049 // An object of this class appends a TestPartResult object to the
00050 // TestPartResultArray object given in the constructor whenever a Google Test
00051 // failure is reported. It can either intercept only failures that are
00052 // generated in the same thread that created this object or it can intercept
00053 // all generated failures. The scope of this mock object can be controlled with
00054 // the second argument to the two arguments constructor.
00055 class GTEST_API_ ScopedFakeTestPartResultReporter
00056     : public TestPartResultReporterInterface {
00057 public:
00058     // The two possible mocking modes of this object.
00059     enum InterceptMode {
00060         INTERCEPT_ONLY_CURRENT_THREAD, // Intercepts only thread local failures.
00061         INTERCEPT_ALL_THREADS        // Intercepts all failures.
00062     };
00063
00064     // The c'tor sets this object as the test part result reporter used
00065     // by Google Test. The 'result' parameter specifies where to report the
00066     // results. This reporter will only catch failures generated in the current
00067     // thread. DEPRECATED
00068     explicit ScopedFakeTestPartResultReporter(TestPartResultArray* result);
00069
00070     // Same as above, but you can choose the interception scope of this object.
00071     ScopedFakeTestPartResultReporter(InterceptMode intercept_mode,
00072                                     TestPartResultArray* result);
00073
00074     // The d'tor restores the previous test part result reporter.
00075     virtual ~ScopedFakeTestPartResultReporter();
00076
00077     // Appends the TestPartResult object to the TestPartResultArray
00078     // received in the constructor.
00079     //
00080     // This method is from the TestPartResultReporterInterface
00081     // interface.
00082     virtual void ReportTestPartResult(const TestPartResult& result);
00083 private:
00084     void Init();
00085
00086     const InterceptMode intercept_mode_;
00087     TestPartResultReporterInterface* old_reporter_;
00088     TestPartResultArray* const result_;
00089
00090     GTEST_DISALLOW_COPY_AND_ASSIGN_(ScopedFakeTestPartResultReporter);
00091 };
00092
00093 namespace internal {
00094
00095 // A helper class for implementing EXPECT_FATAL_FAILURE() and
00096 // EXPECT_NONFATAL_FAILURE(). Its destructor verifies that the given
00097 // TestPartResultArray contains exactly one failure that has the given
00098 // type and contains the given substring. If that's not the case, a
00099 // non-fatal failure will be generated.
00100 class GTEST_API_ SingleFailureChecker {
00101 public:
00102     // The constructor remembers the arguments.
00103     SingleFailureChecker(const TestPartResultArray* results,
00104                           TestPartResult::Type type, const std::string& substr);
00105     ~SingleFailureChecker();
00106 private:
00107     const TestPartResultArray* const results_;
00108     const TestPartResult::Type type_;
00109     const std::string substr_;
00110
00111     GTEST_DISALLOW_COPY_AND_ASSIGN_(SingleFailureChecker);
00112 };
00113
00114 } // namespace internal
00115
00116 } // namespace testing
00117
00118 GTEST_DISABLE_MSC_WARNINGS_POP_() // 4251
00119
00120 // A set of macros for testing Google Test assertions or code that's expected
00121 // to generate Google Test fatal failures. It verifies that the given
00122 // statement will cause exactly one fatal Google Test failure with 'substr'
00123 // being part of the failure message.
00124 //
00125 // There are two different versions of this macro. EXPECT_FATAL_FAILURE only
00126 // affects and considers failures generated in the current thread and
```

```

00127 // EXPECT_FATAL_FAILURE_ON_ALL_THREADS does the same but for all threads.
00128 //
00129 // The verification of the assertion is done correctly even when the statement
00130 // throws an exception or aborts the current function.
00131 //
00132 // Known restrictions:
00133 //   - 'statement' cannot reference local non-static variables or
00134 //     non-static members of the current object.
00135 //   - 'statement' cannot return a value.
00136 //   - You cannot stream a failure message to this macro.
00137 //
00138 // Note that even though the implementations of the following two
00139 // macros are much alike, we cannot refactor them to use a common
00140 // helper macro, due to some peculiarity in how the preprocessor
00141 // works. The AcceptsMacroThatExpandsToUnprotectedComma test in
00142 // gtest_unittest.cc will fail to compile if we do that.
00143 #define EXPECT_FATAL_FAILURE(statement, substr) \
00144 do { \
00145     class GTestExpectFatalFailureHelper { \
00146         public: \
00147             static void Execute() { statement; } \
00148     }; \
00149     ::testing::TestPartResultArray gtest_failures; \
00150     ::testing::internal::SingleFailureChecker gtest_checker( \
00151         &gtest_failures, ::testing::TestPartResult::kFatalFailure, (substr)); \
00152     { \
00153         ::testing::ScopedFakeTestPartResultReporter gtest_reporter( \
00154             ::testing::ScopedFakeTestPartResultReporter:: \
00155             INTERCEPT_ONLY_CURRENT_THREAD, &gtest_failures); \
00156         GTestExpectFatalFailureHelper::Execute(); \
00157     } \
00158 } while (::testing::internal::AlwaysFalse())
00159
00160 #define EXPECT_FATAL_FAILURE_ON_ALL_THREADS(statement, substr) \
00161 do { \
00162     class GTestExpectFatalFailureHelper { \
00163         public: \
00164             static void Execute() { statement; } \
00165     }; \
00166     ::testing::TestPartResultArray gtest_failures; \
00167     ::testing::internal::SingleFailureChecker gtest_checker( \
00168         &gtest_failures, ::testing::TestPartResult::kFatalFailure, (substr)); \
00169     { \
00170         ::testing::ScopedFakeTestPartResultReporter gtest_reporter( \
00171             ::testing::ScopedFakeTestPartResultReporter:: \
00172             INTERCEPT_ALL_THREADS, &gtest_failures); \
00173         GTestExpectFatalFailureHelper::Execute(); \
00174     } \
00175 } while (::testing::internal::AlwaysFalse())
00176
00177 // A macro for testing Google Test assertions or code that's expected to
00178 // generate Google Test non-fatal failures. It asserts that the given
00179 // statement will cause exactly one non-fatal Google Test failure with 'substr'
00180 // being part of the failure message.
00181 //
00182 // There are two different versions of this macro. EXPECT_NONFATAL_FAILURE only
00183 // affects and considers failures generated in the current thread and
00184 // EXPECT_NONFATAL_FAILURE_ON_ALL_THREADS does the same but for all threads.
00185 //
00186 // 'statement' is allowed to reference local variables and members of
00187 // the current object.
00188 //
00189 // The verification of the assertion is done correctly even when the statement
00190 // throws an exception or aborts the current function.
00191 //
00192 // Known restrictions:
00193 //   - You cannot stream a failure message to this macro.
00194 //
00195 // Note that even though the implementations of the following two
00196 // macros are much alike, we cannot refactor them to use a common
00197 // helper macro, due to some peculiarity in how the preprocessor
00198 // works. If we do that, the code won't compile when the user gives
00199 // EXPECT_NONFATAL_FAILURE() a statement that contains a macro that
00200 // expands to code containing an unprotected comma. The
00201 // AcceptsMacroThatExpandsToUnprotectedComma test in gtest_unittest.cc
00202 // catches that.
00203 //
00204 // For the same reason, we have to write
00205 //   if (::testing::internal::AlwaysTrue()) { statement; }
00206 // instead of
00207 //   GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement)
00208 // to avoid an MSVC warning on unreachable code.
00209 #define EXPECT_NONFATAL_FAILURE(statement, substr) \
00210 do { \
00211     ::testing::TestPartResultArray gtest_failures; \
00212     ::testing::internal::SingleFailureChecker gtest_checker( \
00213         &gtest_failures, ::testing::TestPartResult::kNonFatalFailure, \

```

```
00214     (substr)); \
00215     {\ \
00216         ::testing::ScopedFakeTestPartResultReporter gtest_reporter(\ \
00217             ::testing::ScopedFakeTestPartResultReporter:: \
00218             INTERCEPT_ONLY_CURRENT_THREAD, &gtest_failures); \
00219         if (::testing::internal::AlwaysTrue()) { statement; } \
00220     }\ \
00221 } while (::testing::internal::AlwaysFalse())
00222
00223 #define EXPECT_NONFATAL_FAILURE_ON_ALL_THREADS(statement, substr) \
00224 do {\ \
00225     ::testing::TestPartResultArray gtest_failures; \
00226     ::testing::internal::SingleFailureChecker gtest_checker(\ \
00227         &gtest_failures, ::testing::TestPartResult::kNonFatalFailure, \
00228         (substr)); \
00229     {\ \
00230         ::testing::ScopedFakeTestPartResultReporter gtest_reporter(\ \
00231             ::testing::ScopedFakeTestPartResultReporter::INTERCEPT_ALL_THREADS, \
00232             &gtest_failures); \
00233         if (::testing::internal::AlwaysTrue()) { statement; } \
00234     }\ \
00235 } while (::testing::internal::AlwaysFalse())
00236
00237 #endif // GTEST_INCLUDE_GTEST_GTEST_SPI_H
```

9.16 Dokumentacja pliku

packages/[**Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-test-part.h**](#)

```
#include <iostream>
#include <vector>
#include "gtest/internal/gtest-internal.h"
#include "gtest/internal/gtest-string.h"
```

Funkcje

- [**GTEST_DISABLE_MSC_WARNINGS_PUSH_**](#) (4251) namespace **testing**

9.16.1 Dokumentacja funkcji

9.16.1.1 GTEST_DISABLE_MSC_WARNINGS_PUSH_()

```
GTEST_DISABLE_MSC_WARNINGS_PUSH_ (
    4251 )
```

9.17 gtest-test-part.h

[Idź do dokumentacji tego pliku.](#)

```
00001 // Copyright 2008, Google Inc.
00002 // All rights reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //      * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
```

```

00010 //      * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 // GOOGLETEST_CM0001 DO NOT DELETE
00031
00032 #ifndef GTEST_INCLUDE_GTEST_GTEST_TEST_PART_H_
00033 #define GTEST_INCLUDE_GTEST_GTEST_TEST_PART_H_
00034
00035 #include <iostream>
00036 #include <vector>
00037 #include "gtest/internal/gtest-internal.h"
00038 #include "gtest/internal/gtest-string.h"
00039
00040 GTEST_DISABLE_MSC_WARNINGS_PUSH_(4251 \
00041 /* class A needs to have dll-interface to be used by clients of class B */)
00042
00043 namespace testing {
00044
00045 // A copyable object representing the result of a test part (i.e. an
00046 // assertion or an explicit FAIL(), ADD_FAILURE(), or SUCCESS()).
00047 //
00048 // Don't inherit from TestPartResult as its destructor is not virtual.
00049 class GTEST_API_ TestPartResult {
00050 public:
00051     // The possible outcomes of a test part (i.e. an assertion or an
00052     // explicit SUCCEED(), FAIL(), or ADD_FAILURE()).
00053     enum Type {
00054         kSuccess,           // Succeeded.
00055         kNonFatalFailure,  // Failed but the test can continue.
00056         kFatalFailure       // Failed and the test should be terminated.
00057     };
00058
00059     // C'tor. TestPartResult does NOT have a default constructor.
00060     // Always use this constructor (with parameters) to create a
00061     // TestPartResult object.
00062     TestPartResult(Type a_type,
00063                     const char* a_file_name,
00064                     int a_line_number,
00065                     const char* a_message)
00066         : type_(a_type),
00067           file_name_(a_file_name == NULL ? "" : a_file_name),
00068           line_number_(a_line_number),
00069           summary_(ExtractSummary(a_message)),
00070           message_(a_message) {
00071     }
00072
00073     // Gets the outcome of the test part.
00074     Type type() const { return type_; }
00075
00076     // Gets the name of the source file where the test part took place, or
00077     // NULL if it's unknown.
00078     const char* file_name() const {
00079         return file_name_.empty() ? NULL : file_name_.c_str();
00080     }
00081
00082     // Gets the line in the source file where the test part took place,
00083     // or -1 if it's unknown.
00084     int line_number() const { return line_number_; }
00085
00086     // Gets the summary of the failure message.
00087     const char* summary() const { return summary_.c_str(); }
00088
00089     // Gets the message associated with the test part.
00090     const char* message() const { return message_.c_str(); }
00091
00092     // Returns true iff the test part passed.
00093     bool passed() const { return type_ == kSuccess; }
00094
00095     // Returns true iff the test part failed.
00096     bool failed() const { return type_ != kSuccess; }

```

```
00097
00098 // Returns true iff the test part non-fatally failed.
00099 bool nonfatally_failed() const { return type_ == kNonFatalFailure; }
00100
00101 // Returns true iff the test part fatally failed.
00102 bool fatally_failed() const { return type_ == kFatalFailure; }
00103
00104 private:
00105     Type type_;
00106
00107 // Gets the summary of the failure message by omitting the stack
00108 // trace in it.
00109     static std::string ExtractSummary(const char* message);
00110
00111 // The name of the source file where the test part took place, or
00112 // "" if the source file is unknown.
00113     std::string file_name_;
00114 // The line in the source file where the test part took place, or -1
00115 // if the line number is unknown.
00116     int line_number_;
00117     std::string summary_; // The test failure summary.
00118     std::string message_; // The test failure message.
00119 };
00120
00121 // Prints a TestPartResult object.
00122 std::ostream& operator<<(std::ostream& os, const TestPartResult& result);
00123
00124 // An array of TestPartResult objects.
00125 //
00126 // Don't inherit from TestPartResultArray as its destructor is not
00127 // virtual.
00128 class GTEST_API_ TestPartResultArray {
00129 public:
00130     TestPartResultArray() {}
00131
00132 // Appends the given TestPartResult to the array.
00133     void Append(const TestPartResult& result);
00134
00135 // Returns the TestPartResult at the given index (0-based).
00136     const TestPartResult& GetTestPartResult(int index) const;
00137
00138 // Returns the number of TestPartResult objects in the array.
00139     int size() const;
00140
00141 private:
00142     std::vector<TestPartResult> array_;
00143
00144     GTEST_DISALLOW_COPY_AND_ASSIGN_(TestPartResultArray);
00145 };
00146
00147 // This interface knows how to report a test part result.
00148 class GTEST_API_ TestPartResultReporterInterface {
00149 public:
00150     virtual ~TestPartResultReporterInterface() {}
00151
00152     virtual void ReportTestPartResult(const TestPartResult& result) = 0;
00153 };
00154
00155 namespace internal {
00156
00157 // This helper class is used by {ASSERT|EXPECT}_NO_FATAL_FAILURE to check if a
00158 // statement generates new fatal failures. To do so it registers itself as the
00159 // current test part result reporter. Besides checking if fatal failures were
00160 // reported, it only delegates the reporting to the former result reporter.
00161 // The original result reporter is restored in the destructor.
00162 // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
00163 class GTEST_API_ HasNewFatalFailureHelper
00164     : public TestPartResultReporterInterface {
00165 public:
00166     HasNewFatalFailureHelper();
00167     virtual ~HasNewFatalFailureHelper();
00168     virtual void ReportTestPartResult(const TestPartResult& result);
00169     bool has_new_fatal_failure() const { return has_new_fatal_failure_; }
00170 private:
00171     bool has_new_fatal_failure_;
00172     TestPartResultReporterInterface* original_reporter_;
00173
00174     GTEST_DISALLOW_COPY_AND_ASSIGN_(HasNewFatalFailureHelper);
00175 };
00176
00177 } // namespace internal
00178
00179 } // namespace testing
00180
00181 GTEST_DISABLE_MSC_WARNINGS_POP_() // 4251
00182
00183 #endif // GTEST_INCLUDE_GTEST_GTEST_TEST_PART_H_
```

9.18 Dokumentacja pliku

packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-typed-test.h

```
#include "gtest/internal/gtest-port.h"
#include "gtest/internal/gtest-type-util.h"
```

9.19 gtest-typed-test.h

[Idź do dokumentacji tego pliku.](#)

```
00001 // Copyright 2008 Google Inc.
00002 // All Rights Reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //      * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //      * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029
00030
00031 // GOOGLETEST_CM0001 DO NOT DELETE
00032
00033 #ifndef GTEST_INCLUDE_GTEST_GTEST_TYPED_TEST_H_
00034 #define GTEST_INCLUDE_GTEST_GTEST_TYPED_TEST_H_
00035
00036 // This header implements typed tests and type-parameterized tests.
00037
00038 // Typed (aka type-driven) tests repeat the same test for types in a
00039 // list. You must know which types you want to test with when writing
00040 // typed tests. Here's how you do it:
00041
00042 #if 0
00043
00044 // First, define a fixture class template. It should be parameterized
00045 // by a type. Remember to derive it from testing::Test.
00046 template <typename T>
00047 class FooTest : public testing::Test {
00048 public:
00049 ...
00050     typedef std::list<T> List;
00051     static T shared_;
00052     T value_;
00053 };
00054
00055 // Next, associate a list of types with the test case, which will be
00056 // repeated for each type in the list. The typedef is necessary for
00057 // the macro to parse correctly.
00058 typedef testing::Types<char, int, unsigned int> MyTypes;
00059 TYPED_TEST_CASE(FooTest, MyTypes);
00060
00061 // If the type list contains only one type, you can write that type
00062 // directly without Types<...>:
00063 //    TYPED_TEST_CASE(FooTest, int);
00064
```

```
00065 // Then, use TYPED_TEST() instead of TEST_F() to define as many typed
00066 // tests for this test case as you want.
00067 TYPED_TEST(FooTest, DoesBlah) {
00068     // Inside a test, refer to TypeParam to get the type parameter.
00069     // Since we are inside a derived class template, C++ requires use to
00070     // visit the members of FooTest via 'this'.
00071     TypeParam n = this->value_;
00072
00073     // To visit static members of the fixture, add the TestFixture::
00074     // prefix.
00075     n += TestFixture::shared_;
00076
00077     // To refer to typedefs in the fixture, add the "typename
00078     // TestFixture::" prefix.
00079     typename TestFixture::List values;
00080     values.push_back(n);
00081     ...
00082 }
00083
00084 TYPED_TEST(FooTest, HasPropertyA) { ... }
00085
00086 // TYPED_TEST_CASE takes an optional third argument which allows to specify a
00087 // class that generates custom test name suffixes based on the type. This should
00088 // be a class which has a static template function GetName(int index) returning
00089 // a string for each type. The provided integer index equals the index of the
00090 // type in the provided type list. In many cases the index can be ignored.
00091 //
00092 // For example:
00093 //   class MyTypeNames {
00094 //     public:
00095 //       template <typename T>
00096 //         static std::string GetName(int) {
00097 //           if (std::is_same<T, char>()) return "char";
00098 //           if (std::is_same<T, int>()) return "int";
00099 //           if (std::is_same<T, unsigned int>()) return "unsignedInt";
00100 //         }
00101 //   };
00102 //   TYPED_TEST_CASE(FooTest, MyTypes, MyTypeNames);
00103
00104 #endif // 0
00105
00106 // Type-parameterized tests are abstract test patterns parameterized
00107 // by a type. Compared with typed tests, type-parameterized tests
00108 // allow you to define the test pattern without knowing what the type
00109 // parameters are. The defined pattern can be instantiated with
00110 // different types any number of times, in any number of translation
00111 // units.
00112 //
00113 // If you are designing an interface or concept, you can define a
00114 // suite of type-parameterized tests to verify properties that any
00115 // valid implementation of the interface/concept should have. Then,
00116 // each implementation can easily instantiate the test suite to verify
00117 // that it conforms to the requirements, without having to write
00118 // similar tests repeatedly. Here's an example:
00119
00120 #if 0
00121
00122 // First, define a fixture class template. It should be parameterized
00123 // by a type. Remember to derive it from testing::Test.
00124 template <typename T>
00125 class FooTest : public testing::Test {
00126     ...
00127 };
00128
00129 // Next, declare that you will define a type-parameterized test case
00130 // (the _P suffix is for "parameterized" or "pattern", whichever you
00131 // prefer):
00132 TYPED_TEST_CASE_P(FooTest);
00133
00134 // Then, use TYPED_TEST_P() to define as many type-parameterized tests
00135 // for this type-parameterized test case as you want.
00136 TYPED_TEST_P(FooTest, DoesBlah) {
00137     // Inside a test, refer to TypeParam to get the type parameter.
00138     TypeParam n = 0;
00139     ...
00140 }
00141
00142 TYPED_TEST_P(FooTest, HasPropertyA) { ... }
00143
00144 // Now the tricky part: you need to register all test patterns before
00145 // you can instantiate them. The first argument of the macro is the
00146 // test case name; the rest are the names of the tests in this test
00147 // case.
00148 REGISTER_TYPED_TEST_CASE_P(FooTest,
00149                         DoesBlah, HasPropertyA);
00150
00151 // Finally, you are free to instantiate the pattern with the types you
```

```

00152 // want. If you put the above code in a header file, you can #include
00153 // it in multiple C++ source files and instantiate it multiple times.
00154 //
00155 // To distinguish different instances of the pattern, the first
00156 // argument to the INSTANTIATE_* macro is a prefix that will be added
00157 // to the actual test case name. Remember to pick unique prefixes for
00158 // different instances.
00159 typedef testing::Types<char, int, unsigned int> MyTypes;
00160 INSTANTIATE_TYPED_TEST_CASE_P(My, FooTest, MyTypes);
00161
00162 // If the type list contains only one type, you can write that type
00163 // directly without Types<...>:
00164 // INSTANTIATE_TYPED_TEST_CASE_P(My, FooTest, int);
00165 //
00166 // Similar to the optional argument of TYPED_TEST_CASE above,
00167 // INSTANTIATE_TEST_CASE_P takes an optional fourth argument which allows to
00168 // generate custom names.
00169 // INSTANTIATE_TYPED_TEST_CASE_P(My, FooTest, MyTypes, MyTypeNames);
00170
00171 #endif // 0
00172
00173 #include "gtest/internal/gtest-port.h"
00174 #include "gtest/internal/gtest-type-util.h"
00175
00176 // Implements typed tests.
00177
00178 #if GTEST_HAS_TYPED_TEST
00179
00180 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
00181 //
00182 // Expands to the name of the typedef for the type parameters of the
00183 // given test case.
00184 #define GTEST_TYPE_PARAMS_(TestCaseName) gtest_type_params_##TestCaseName##_
00185
00186 // Expands to the name of the typedef for the NameGenerator, responsible for
00187 // creating the suffixes of the name.
00188 #define GTEST_NAME_GENERATOR_(TestCaseName) \
00189     gtest_type_params_##TestCaseName##_NameGenerator
00190
00191 // The 'Types' template argument below must have spaces around it
00192 // since some compilers may choke on '' when passing a template
00193 // instance (e.g. Types<int>)
00194 #define TYPED_TEST_CASE(CaseName, Types, ...)
00195     typedef ::testing::internal::TypeList< Types >::type GTEST_TYPE_PARAMS_( \
00196         CaseName); \
00197     typedef ::testing::internal::NameGeneratorSelector<__VA_ARGS__>::type \
00198         GTEST_NAME_GENERATOR_(CaseName)
00199
00200 #define TYPED_TEST(CaseName, TestName)
00201     template <typename gtest_TypeParam>
00202     class GTEST_TEST_CLASS_NAME_(CaseName, TestName)
00203     : public CaseName<gtest_TypeParam> {
00204     private:
00205         typedef CaseName<gtest_TypeParam> TestFixture;
00206         typedef gtest_TypeParam TypeParam;
00207         virtual void TestBody();
00208     };
00209     static bool gtest_##CaseName##_##TestName##_registered_
00210         GTEST_ATTRIBUTE_UNUSED_ =
00211             ::testing::internal::TypeParameterizedTest<
00212                 CaseName,
00213                 ::testing::internal::TemplateSel<GTEST_TEST_CLASS_NAME_(CaseName,
00214                     TestName)>,
00215                 GTEST_TYPE_PARAMS_( \
00216                     CaseName)>::Register("");
00217             ::testing::internal::CodeLocation( \
00218                 __FILE__, __LINE__),
00219                 #CaseName, #TestName, 0,
00220                 ::testing::internal::GenerateNames<
00221                     GTEST_NAME_GENERATOR_(CaseName),
00222                     GTEST_TYPE_PARAMS_(CaseName)>());
00223     template <typename gtest_TypeParam>
00224     void GTEST_TEST_CLASS_NAME_(CaseName,
00225         TestName)<gtest_TypeParam>::TestBody()
00226
00227 #endif // GTEST_HAS_TYPED_TEST
00228
00229 // Implements type-parameterized tests.
00230
00231 #if GTEST_HAS_TYPED_TEST_P
00232
00233 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
00234 //
00235 // Expands to the namespace name that the type-parameterized tests for
00236 // the given type-parameterized test case are defined in. The exact
00237 // name of the namespace is subject to change without notice.
00238 #define GTEST_CASE_NAMESPACE_(TestCaseName) \

```

```
00239     gtest_case_##TestCaseName##_
00240
00241 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
00242 //
00243 // Expands to the name of the variable used to remember the names of
00244 // the defined tests in the given test case.
00245 # define GTEST_TYPED_TEST_CASE_P_STATE_(TestCaseName) \
00246     gtest_typed_test_case_p_state_##TestCaseName##_
00247
00248 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE DIRECTLY.
00249 //
00250 // Expands to the name of the variable used to remember the names of
00251 // the registered tests in the given test case.
00252 # define GTEST_REGISTERED_TEST_NAMES_(TestCaseName) \
00253     gtest_registered_test_names_##TestCaseName##_
00254
00255 // The variables defined in the type-parameterized test macros are
00256 // static as typically these macros are used in a .h file that can be
00257 // #included in multiple translation units linked together.
00258 # define TYPED_TEST_CASE_P(CaseName) \
00259     static ::testing::internal::TypedTestCasePState \
00260         GTEST_TYPED_TEST_CASE_P_STATE_(CaseName)
00261
00262 # define TYPED_TEST_P(CaseName, TestName) \
00263     namespace GTEST_CASE_NAMESPACE_(CaseName) { \
00264         template <typename gtest_TypeParam> \
00265             class TestName : public CaseName<gtest_TypeParam> { \
00266                 private: \
00267                     typedef CaseName<gtest_TypeParam> TestFixture; \
00268                     typedef gtest_TypeParam TypeParam; \
00269                     virtual void TestBody(); \
00270             }; \
00271         static bool gtest_##TestName##_defined_ GTEST_ATTRIBUTE_UNUSED_ = \
00272             GTEST_TYPED_TEST_CASE_P_STATE_(CaseName).AddTestName( \
00273                 __FILE__, __LINE__, #CaseName, #TestName); \
00274     } \
00275     template <typename gtest_TypeParam> \
00276         void GTEST_CASE_NAMESPACE_(CaseName)::TestName<gtest_TypeParam>::TestBody()
00277
00278 # define REGISTER_TYPED_TEST_CASE_P(CaseName, ...) \
00279     namespace GTEST_CASE_NAMESPACE_(CaseName) { \
00280         typedef ::testing::internal::Templates<__VA_ARGS__>::type gtest_AllTests_; \
00281     } \
00282     static const char* const GTEST_REGISTERED_TEST_NAMES_(CaseName) \
00283         GTEST_ATTRIBUTE_UNUSED_ = \
00284             GTEST_TYPED_TEST_CASE_P_STATE_(CaseName).VerifyRegisteredTestNames( \
00285                 __FILE__, __LINE__, #__VA_ARGS__)
00286
00287 // The 'Types' template argument below must have spaces around it
00288 // since some compilers may choke on '」' when passing a template
00289 // instance (e.g. Types<int>)
00290 # define INSTANTIATE_TYPED_TEST_CASE_P(Prefix, CaseName, Types, ...) \
00291     static bool gtest_##Prefix##_##CaseName GTEST_ATTRIBUTE_UNUSED_ = \
00292         ::testing::internal::TypeParameterizedTestCase< \
00293             CaseName, GTEST_CASE_NAMESPACE_(CaseName)::gtest_AllTests_, \
00294             ::testing::internal::TypeList< Types >::type>:: \
00295             Register(#Prefix, \
00296                 ::testing::internal::CodeLocation(__FILE__, __LINE__), \
00297                 &GTEST_TYPED_TEST_CASE_P_STATE_(CaseName), #CaseName, \
00298                 GTEST_REGISTERED_TEST_NAMES_(CaseName), \
00299                 ::testing::internal::GenerateNames< \
00300                     ::testing::internal::NameGeneratorSelector< \
00301                         __VA_ARGS__>::type, \
00302                     ::testing::internal::TypeList< Types >::type>() )
00303
00304 #endif // GTEST_HAS_TYPED_TEST_P
00305
00306 #endif // GTEST_INCLUDE_GTEST_GTEST_TYPED_TEST_H_
```

9.20 Dokumentacja pliku packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest.h

```
#include <limits>
#include <iostream>
#include <vector>
#include "gtest/internal/gtest-internal.h"
```

```
#include "gtest/internal/gtest-string.h"
#include "gtest/gtest-death-test.h"
#include "gtest/gtest-message.h"
#include "gtest/gtest-param-test.h"
#include "gtest/gtest-printers.h"
#include "gtest/gtest_prod.h"
#include "gtest/gtest-test-part.h"
#include "gtest/gtest-typed-test.h"
#include "gtest/gtest_pred_impl.h"
```

Komponenty

- class `testing::Test`
- class `testing::TestProperty`
- class `testing::TestResult`
- class `testing::TestInfo`
- class `testing::TestCase`
- class `testing::Environment`
- class `testing::TestEventListener`
- class `testing::EmptyTestEventListener`
- class `testing::TestEventListeners`
- class `testing::UnitTest`
- class `testing::internal::EqHelper< lhs_is_null_literal >`
- class `testing::internal::EqHelper< true >`
- class `testing::internal::AssertHelper`
- class `testing::WithParamInterface< T >`
- class `testing::TestWithParam< T >`
- class `testing::ScopedTrace`

Przestrzenie nazw

- namespace `testing`
- namespace `testing::internal`

Definicje

- `#define GTEST_IMPL_CMP_HELPER_(op_name, op)`
- `#define ADD_FAILURE()`
- `#define ADD_FAILURE_AT(file, line)`
- `#define GTEST_FAIL()`
- `#define FAIL()`
- `#define GTEST_SUCCEED()`
- `#define SUCCEED()`
- `#define EXPECT_THROW(statement, expected_exception)`
- `#define EXPECT_NO_THROW(statement)`
- `#define EXPECT_ANY_THROW(statement)`
- `#define ASSERT_THROW(statement, expected_exception)`
- `#define ASSERT_NO_THROW(statement)`
- `#define ASSERT_ANY_THROW(statement)`
- `#define EXPECT_TRUE(condition)`
- `#define EXPECT_FALSE(condition)`
- `#define ASSERT_TRUE(condition)`

- `#define ASSERT_FALSE(condition)`
- `#define EXPECT_EQ(val1, val2)`
- `#define EXPECT_NE(val1, val2)`
- `#define EXPECT_LE(val1, val2)`
- `#define EXPECT_LT(val1, val2)`
- `#define EXPECT_GE(val1, val2)`
- `#define EXPECT_GT(val1, val2)`
- `#define GTEST_ASSERT_EQ(val1, val2)`
- `#define GTEST_ASSERT_NE(val1, val2)`
- `#define GTEST_ASSERT_LE(val1, val2)`
- `#define GTEST_ASSERT_LT(val1, val2)`
- `#define GTEST_ASSERT_GE(val1, val2)`
- `#define GTEST_ASSERT_GT(val1, val2)`
- `#define ASSERT_EQ(val1, val2)`
- `#define ASSERT_NE(val1, val2)`
- `#define ASSERT_LE(val1, val2)`
- `#define ASSERT_LT(val1, val2)`
- `#define ASSERT_GE(val1, val2)`
- `#define ASSERT_GT(val1, val2)`
- `#define EXPECT_STREQ(s1, s2)`
- `#define EXPECT_STRNE(s1, s2)`
- `#define EXPECT_STRCASEEQ(s1, s2)`
- `#define EXPECT_STRCASENE(s1, s2)`
- `#define ASSERT_STREQ(s1, s2)`
- `#define ASSERT_STRNE(s1, s2)`
- `#define ASSERT_STRCASEEQ(s1, s2)`
- `#define ASSERT_STRCASENE(s1, s2)`
- `#define EXPECT_FLOAT_EQ(val1, val2)`
- `#define EXPECT_DOUBLE_EQ(val1, val2)`
- `#define ASSERT_FLOAT_EQ(val1, val2)`
- `#define ASSERT_DOUBLE_EQ(val1, val2)`
- `#define EXPECT_NEAR(val1, val2, abs_error)`
- `#define ASSERT_NEAR(val1, val2, abs_error)`
- `#define ASSERT_NO_FATAL_FAILURE(statement)`
- `#define EXPECT_NO_FATAL_FAILURE(statement)`
- `#define SCOPED_TRACE(message)`
- `#define GTEST_TEST(test_case_name, test_name)`
- `#define TEST(test_case_name, test_name)`
- `#define TEST_F(test_fixture, test_name)`

Definicje typów

- `typedef internal::TimeInMillis testing::TimeInMillis`

Funkcje

- `GTEST_DISABLE_MSC_WARNINGS_PUSH_(4251) namespace testing`
- `Environment * testing::AddGlobalTestEnvironment (Environment *env)`
- `GTEST_API_ void testing::InitGoogleTest (int *argc, char **argv)`
- `GTEST_API_ void testing::InitGoogleTest (int *argc, wchar_t **argv)`
- template<typename T1, typename T2>
`AssertionResult testing::internal::CmpHelperEQFailure (const char *lhs_expression, const char *rhs_expression, const T1 &lhs, const T2 &rhs)`

- template<typename T1, typename T2>
AssertionResult [testing::internal::CmpHelperEQ](#) (const char *lhs_expression, const char *rhs_expression, const T1 &lhs, const T2 &rhs)
- [GTEST_API_ AssertionResult testing::internal::CmpHelperEQ](#) (const char *lhs_expression, const char *rhs_expression, BiggestInt lhs, BiggestInt rhs)
- template<typename T1, typename T2>
AssertionResult [testing::internal::CmpHelperOpFailure](#) (const char *expr1, const char *expr2, const T1 &val1, const T2 &val2, const char *op)
- [testing::internal::GTEST_IMPL_CMP_HELPER_\(NE, !=\)](#)
- [testing::internal::GTEST_IMPL_CMP_HELPER_\(LE,<=\)](#)
- [testing::internal::GTEST_IMPL_CMP_HELPER_\(LT,<\)](#)
- [testing::internal::GTEST_IMPL_CMP_HELPER_\(GE,>=\)](#)
- [testing::internal::GTEST_IMPL_CMP_HELPER_\(GT,>\)](#)
- [GTEST_API_ AssertionResult testing::internal::CmpHelperSTREQ](#) (const char *s1_expression, const char *s2_expression, const char *s1, const char *s2)
- [GTEST_API_ AssertionResult testing::internal::CmpHelperSTRCASEEQ](#) (const char *s1_expression, const char *s2_expression, const char *s1, const char *s2)
- [GTEST_API_ AssertionResult testing::internal::CmpHelperSTRNE](#) (const char *s1_expression, const char *s2_expression, const char *s1, const char *s2)
- [GTEST_API_ AssertionResult testing::internal::CmpHelperSTRCASENE](#) (const char *s1_expression, const char *s2_expression, const char *s1, const char *s2)
- [GTEST_API_ AssertionResult testing::internal::CmpHelperSTREQ](#) (const char *s1_expression, const char *s2_expression, const wchar_t *s1, const wchar_t *s2)
- [GTEST_API_ AssertionResult testing::internal::CmpHelperSTRNE](#) (const char *s1_expression, const char *s2_expression, const wchar_t *s1, const wchar_t *s2)
- [GTEST_API_ AssertionResult testing::internal::IsSubstring](#) (const char *needle_expr, const char *haystack_expr, const char *needle, const char *haystack)
- [GTEST_API_ AssertionResult testing::internal::IsSubstring](#) (const char *needle_expr, const char *haystack_expr, const wchar_t *needle, const wchar_t *haystack)
- [GTEST_API_ AssertionResult testing::internal::IsNotSubstring](#) (const char *needle_expr, const char *haystack_expr, const char *needle, const char *haystack)
- [GTEST_API_ AssertionResult testing::internal::IsNotSubstring](#) (const char *needle_expr, const char *haystack_expr, const wchar_t *needle, const wchar_t *haystack)
- [GTEST_API_ AssertionResult testing::internal::IsSubstring](#) (const char *needle_expr, const char *haystack_expr, const ::std::string &needle, const ::std::string &haystack)
- [GTEST_API_ AssertionResult testing::internal::IsNotSubstring](#) (const char *needle_expr, const char *haystack_expr, const ::std::string &needle, const ::std::string &haystack)
- template<typename RawType>
AssertionResult [testing::internal::CmpHelperFloatingPointEQ](#) (const char *lhs_expression, const char *rhs_expression, RawType lhs_value, RawType rhs_value)
- [GTEST_API_ AssertionResult testing::internal::DoubleNearPredFormat](#) (const char *expr1, const char *expr2, const char *abs_error_expr, double val1, double val2, double abs_error)
- [GTEST_API_ AssertionResult testing::internal::FloatLE](#) (const char *expr1, const char *expr2, float val1, float val2)
- [GTEST_API_ AssertionResult testing::internal::DoubleLE](#) (const char *expr1, const char *expr2, double val1, double val2)
- template<typename T1, typename T2>
bool [testing::StaticAssertTypeEq](#) ()
- [GTEST_API_ std::string testing::TempDir](#) ()
- int [RUN_ALL_TESTS](#) () [GTEST_MUST_USE_RESULT_](#)

Zmienne

- template<typename T>
const T * [testing::WithParamInterface< T >::parameter_](#) = NULL
- class [GTEST_API_ testing::ScopedTrace](#) [GTEST_ATTRIBUTE_UNUSED_](#)

9.20.1 Dokumentacja definicji

9.20.1.1 ADD_FAILURE

```
#define ADD_FAILURE()
```

Wartość:

```
GTEST_NONFATAL_FAILURE_("Failed")
```

9.20.1.2 ADD_FAILURE_AT

```
#define ADD_FAILURE_AT(\
    file, \
    line)
```

Wartość:

```
GTEST_MESSAGE_AT_(file, line, "Failed", \
                   ::testing::TestPartResult::kNonFatalFailure)
```

9.20.1.3 ASSERT_ANY_THROW

```
#define ASSERT_ANY_THROW(\
    statement)
```

Wartość:

```
GTEST_TEST_ANY_THROW_(statement, GTEST_FATAL_FAILURE_)
```

9.20.1.4 ASSERT_DOUBLE_EQ

```
#define ASSERT_DOUBLE_EQ(\
    val1, \
    val2)
```

Wartość:

```
ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperFloatingPointEQ<double>, \
                   val1, val2)
```

9.20.1.5 ASSERT_EQ

```
#define ASSERT_EQ(\
    val1, \
    val2)
```

Wartość:

```
GTEST_ASSERT_EQ(val1, val2)
```

9.20.1.6 ASSERT_FALSE

```
#define ASSERT_FALSE(  
    condition)
```

Wartość:

```
GTEST_TEST_BOOLEAN_(!(condition), #condition, true, false, \  
    GTEST_FATAL_FAILURE_)
```

9.20.1.7 ASSERT_FLOAT_EQ

```
#define ASSERT_FLOAT_EQ(  
    val1,  
    val2)
```

Wartość:

```
ASSERT_PRED_FORMAT2 (:testing::internal::CmpHelperFloatingPointEQ<float>, \  
    val1, val2)
```

9.20.1.8 ASSERT_GE

```
#define ASSERT_GE(  
    val1,  
    val2)
```

Wartość:

```
GTEST_ASSERT_GE(val1, val2)
```

9.20.1.9 ASSERT_GT

```
#define ASSERT_GT(  
    val1,  
    val2)
```

Wartość:

```
GTEST_ASSERT_GT(val1, val2)
```

9.20.1.10 ASSERT_LE

```
#define ASSERT_LE(  
    val1,  
    val2)
```

Wartość:

```
GTEST_ASSERT_LE(val1, val2)
```

9.20.1.11 ASSERT_LT

```
#define ASSERT_LT(  
    val1,  
    val2)
```

Wartość:

```
GTEST_ASSERT_LT(val1, val2)
```

9.20.1.12 ASSERT_NE

```
#define ASSERT_NE(  
    val1,  
    val2)
```

Wartość:

```
GTEST_ASSERT_NE(val1, val2)
```

9.20.1.13 ASSERT_NEAR

```
#define ASSERT_NEAR(  
    val1,  
    val2,  
    abs_error)
```

Wartość:

```
ASSERT_PRED_FORMAT3(::testing::internal::DoubleNearPredFormat, \  
    val1, val2, abs_error)
```

9.20.1.14 ASSERT_NO_FATAL_FAILURE

```
#define ASSERT_NO_FATAL_FAILURE(  
    statement)
```

Wartość:

```
GTEST_TEST_NO_FATAL_FAILURE_(statement, GTEST_FATAL_FAILURE_)
```

9.20.1.15 ASSERT_NO_THROW

```
#define ASSERT_NO_THROW(  
    statement)
```

Wartość:

```
GTEST_TEST_NO_THROW_(statement, GTEST_FATAL_FAILURE_)
```

9.20.1.16 ASSERT_STRCASEEQ

```
#define ASSERT_STRCASEEQ(  
    s1,  
    s2)
```

Wartość:

```
ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperSTRCASEEQ, s1, s2)
```

9.20.1.17 ASSERT_STRCASENE

```
#define ASSERT_STRCASENE(
    s1,
    s2)
```

Wartość:

```
ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperSTRCASENE, s1, s2)
```

9.20.1.18 ASSERT_STREQ

```
#define ASSERT_STREQ(
    s1,
    s2)
```

Wartość:

```
ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperSTREQ, s1, s2)
```

9.20.1.19 ASSERT_STRNE

```
#define ASSERT_STRNE(
    s1,
    s2)
```

Wartość:

```
ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperSTRNE, s1, s2)
```

9.20.1.20 ASSERT_THROW

```
#define ASSERT_THROW(
    statement,
    expected_exception)
```

Wartość:

```
GTEST_TEST_THROW_(statement, expected_exception, GTEST_FATAL_FAILURE_)
```

9.20.1.21 ASSERT_TRUE

```
#define ASSERT_TRUE(
    condition)
```

Wartość:

```
GTEST_TEST_BOOLEAN_(condition, #condition, false, true, \
GTEST_FATAL_FAILURE_)
```

9.20.1.22 EXPECT_ANY_THROW

```
#define EXPECT_ANY_THROW(
    statement)
```

Wartość:

```
GTEST_TEST_ANY_THROW_(statement, GTEST_NONFATAL_FAILURE_)
```

9.20.1.23 EXPECT_DOUBLE_EQ

```
#define EXPECT_DOUBLE_EQ(  
    val1,  
    val2)
```

Wartość:

```
EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperFloatingPointEQ<double>, \  
    val1, val2)
```

9.20.1.24 EXPECT_EQ

```
#define EXPECT_EQ(  
    val1,  
    val2)
```

Wartość:

```
EXPECT_PRED_FORMAT2(::testing::internal::EqHelper<GTEST_IS_NULL_LITERAL_(val1)>::Compare, \  
    val1, val2)
```

9.20.1.25 EXPECT_FALSE

```
#define EXPECT_FALSE(  
    condition)
```

Wartość:

```
GTEST_TEST_BOOLEAN_(!(condition), #condition, true, false, \  
    GTEST_NONFATAL_FAILURE_)
```

9.20.1.26 EXPECT_FLOAT_EQ

```
#define EXPECT_FLOAT_EQ(  
    val1,  
    val2)
```

Wartość:

```
EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperFloatingPointEQ<float>, \  
    val1, val2)
```

9.20.1.27 EXPECT_GE

```
#define EXPECT_GE(  
    val1,  
    val2)
```

Wartość:

```
EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperGE, val1, val2)
```

9.20.1.28 EXPECT_GT

```
#define EXPECT_GT(  
    val1,  
    val2)
```

Wartość:

```
EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperGT, val1, val2)
```

9.20.1.29 EXPECT_LE

```
#define EXPECT_LE(  
    val1,  
    val2)
```

Wartość:

```
EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperLE, val1, val2)
```

9.20.1.30 EXPECT_LT

```
#define EXPECT_LT(  
    val1,  
    val2)
```

Wartość:

```
EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperLT, val1, val2)
```

9.20.1.31 EXPECT_NE

```
#define EXPECT_NE(  
    val1,  
    val2)
```

Wartość:

```
EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperNE, val1, val2)
```

9.20.1.32 EXPECT_NEAR

```
#define EXPECT_NEAR(  
    val1,  
    val2,  
    abs_error)
```

Wartość:

```
EXPECT_PRED_FORMAT3(::testing::internal::DoubleNearPredFormat, \  
    val1, val2, abs_error)
```

9.20.1.33 EXPECT_NO_FATAL_FAILURE

```
#define EXPECT_NO_FATAL_FAILURE(  
    statement)
```

Wartość:

```
GTEST_TEST_NO_FATAL_FAILURE_(statement, GTEST_NONFATAL_FAILURE_)
```

9.20.1.34 EXPECT_NO_THROW

```
#define EXPECT_NO_THROW(  
    statement)
```

Wartość:

```
GTEST_TEST_NO_THROW_(statement, GTEST_NONFATAL_FAILURE_)
```

9.20.1.35 EXPECT_STRCASEEQ

```
#define EXPECT_STRCASEEQ(  
    s1,  
    s2)
```

Wartość:

```
EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperSTRCASEEQ, s1, s2)
```

9.20.1.36 EXPECT_STRCASENE

```
#define EXPECT_STRCASENE(  
    s1,  
    s2)
```

Wartość:

```
EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperSTRCASENE, s1, s2)
```

9.20.1.37 EXPECT_STREQ

```
#define EXPECT_STREQ(  
    s1,  
    s2)
```

Wartość:

```
EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperSTREQ, s1, s2)
```

9.20.1.38 EXPECT_STRNE

```
#define EXPECT_STRNE(  
    s1,  
    s2)
```

Wartość:

```
EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperSTRNE, s1, s2)
```

9.20.1.39 EXPECT_THROW

```
#define EXPECT_THROW(
    statement,
    expected_exception)
```

Wartość:

```
GTEST_TEST_THROW_(statement, expected_exception, GTEST_NONFATAL_FAILURE_)
```

9.20.1.40 EXPECT_TRUE

```
#define EXPECT_TRUE(
    condition)
```

Wartość:

```
GTEST_TEST_BOOLEAN_(condition, #condition, false, true, \
    GTEST_NONFATAL_FAILURE_)
```

9.20.1.41 FAIL

```
#define FAIL()
```

Wartość:

```
GTEST_FAIL()
```

9.20.1.42 GTEST_ASSERT_EQ

```
#define GTEST_ASSERT_EQ(
    val1,
    val2)
```

Wartość:

```
ASSERT_PRED_FORMAT2(:testing::internal:: \
    EqHelper<GTEST_IS_NULL_LITERAL_(val1)>::Compare, \
    val1, val2)
```

9.20.1.43 GTEST_ASSERT_GE

```
#define GTEST_ASSERT_GE(
    val1,
    val2)
```

Wartość:

```
ASSERT_PRED_FORMAT2(:testing::internal::CmpHelperGE, val1, val2)
```

9.20.1.44 GTEST_ASSERT_GT

```
#define GTEST_ASSERT_GT(
    val1,
    val2)
```

Wartość:

```
ASSERT_PRED_FORMAT2(:testing::internal::CmpHelperGT, val1, val2)
```

9.20.1.45 GTEST_ASSERT_LE

```
#define GTEST_ASSERT_LE(\
    val1, \
    val2)
```

Wartość:

```
ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperLE, val1, val2)
```

9.20.1.46 GTEST_ASSERT_LT

```
#define GTEST_ASSERT_LT(\
    val1, \
    val2)
```

Wartość:

```
ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperLT, val1, val2)
```

9.20.1.47 GTEST_ASSERT_NE

```
#define GTEST_ASSERT_NE(\
    val1, \
    val2)
```

Wartość:

```
ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperNE, val1, val2)
```

9.20.1.48 GTEST_FAIL

```
#define GTEST_FAIL()
```

Wartość:

```
GTEST_FATAL_FAILURE_("Failed")
```

9.20.1.49 GTEST_IMPL_CMP_HELPER_

```
#define GTEST_IMPL_CMP_HELPER_(\
    op_name, \
    op) \
template <typename T1, typename T2> \
AssertionResult CmpHelper#op_name(const char* expr1, const char* expr2, \
                                    const T1& val1, const T2& val2) { \
    if (val1 op val2) { \
        return AssertionSuccess(); \
    } else { \
        return CmpHelperOpFailure(expr1, expr2, val1, val2, #op); \
    } \
} \
GTEST_API_ AssertionResult CmpHelper#op_name( \
    const char* expr1, const char* expr2, BiggestInt val1, BiggestInt val2)
```

9.20.1.50 GTEST_SUCCEED

```
#define GTEST_SUCCEED()
```

Wartość:

```
GTEST_SUCCESS_("Succeeded")
```

9.20.1.51 GTEST_TEST

```
#define GTEST_TEST(\
    test_case_name, \
    test_name)
```

Wartość:

```
GTEST_TEST_(test_case_name, test_name, \
    ::testing::Test, ::testing::internal::GetTypeId())
```

9.20.1.52 SCOPED_TRACE

```
#define SCOPED_TRACE(\
    message)
```

Wartość:

```
::testing::ScopedTrace GTEST_CONCAT_TOKEN_(gtest_trace_, __LINE__)(\
    __FILE__, __LINE__, (message))
```

9.20.1.53 SUCCEED

```
#define SUCCEED()
```

Wartość:

```
GTEST_SUCCEED()
```

9.20.1.54 TEST

```
#define TEST(\
    test_case_name, \
    test_name)
```

Wartość:

```
GTEST_TEST(test_case_name, test_name)
```

9.20.1.55 TEST_F

```
#define TEST_F(\
    test_fixture, \
    test_name)
```

Wartość:

```
GTEST_TEST_(test_fixture, test_name, test_fixture, \
    ::testing::internal::GetTypeId<test_fixture>())
```

9.20.2 Dokumentacja funkcji

9.20.2.1 GTEST_DISABLE_MSC_WARNINGS_PUSH_()

```
GTEST_DISABLE_MSC_WARNINGS_PUSH_ (
    4251 )
```

9.20.2.2 RUN_ALL_TESTS()

```
int RUN_ALL_TESTS () [inline]
```

9.21 gtest.h

[Idź do dokumentacji tego pliku.](#)

```
00001 // Copyright 2005, Google Inc.
00002 // All rights reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //      * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //      * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 //
00031 // The Google C++ Testing and Mocking Framework (Google Test)
00032 //
00033 // This header file defines the public API for Google Test. It should be
00034 // included by any test program that uses Google Test.
00035 //
00036 // IMPORTANT NOTE: Due to limitation of the C++ language, we have to
00037 // leave some internal implementation details in this header file.
00038 // They are clearly marked by comments like this:
00039 //
00040 // // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
00041 //
00042 // Such code is NOT meant to be used by a user directly, and is subject
00043 // to CHANGE WITHOUT NOTICE. Therefore DO NOT DEPEND ON IT in a user
00044 // program!
00045 //
00046 // Acknowledgment: Google Test borrowed the idea of automatic test
00047 // registration from Barthelemy Dagenais' (barthelemy@prologue.com)
00048 // easyUnit framework.
00049 //
00050 // GOOGLETEST_CM0001 DO NOT DELETE
00051
00052 #ifndef GTEST_INCLUDE_GTEST_GTEST_H_
00053 #define GTEST_INCLUDE_GTEST_GTEST_H_
00054
00055 #include <limits>
00056 #include <iostream>
00057 #include <vector>
00058
```

```
00059 #include "gtest/internal/gtest-internal.h"
00060 #include "gtest/internal/gtest-string.h"
00061 #include "gtest/gtest-death-test.h"
00062 #include "gtest/gtest-message.h"
00063 #include "gtest/gtest-param-test.h"
00064 #include "gtest/gtest-printers.h"
00065 #include "gtest/gtest_prod.h"
00066 #include "gtest/gtest-test-part.h"
00067 #include "gtest/gtest-typed-test.h"
00068
00069 GTEST_DISABLE_MSC_WARNINGS_PUSH_(4251 \
00070 /* class A needs to have dll-interface to be used by clients of class B */
00071
00072 // Depending on the platform, different string classes are available.
00073 // On Linux, in addition to ::std::string, Google also makes use of
00074 // class ::string, which has the same interface as ::std::string, but
00075 // has a different implementation.
00076 //
00077 // You can define GTEST_HAS_GLOBAL_STRING to 1 to indicate that
00078 // ::string is available AND is a distinct type to ::std::string, or
00079 // define it to 0 to indicate otherwise.
00080 //
00081 // If ::std::string and ::string are the same class on your platform
00082 // due to aliasing, you should define GTEST_HAS_GLOBAL_STRING to 0.
00083 //
00084 // If you do not define GTEST_HAS_GLOBAL_STRING, it is defined
00085 // heuristically.
00086
00087 namespace testing {
00088
00089 // Silence C4100 (unreferenced formal parameter) and 4805
00090 // unsafe mix of type 'const int' and type 'const bool'
00091 #ifdef _MSC_VER
00092 # pragma warning(push)
00093 # pragma warning(disable:4805)
00094 # pragma warning(disable:4100)
00095 #endif
00096
00097
00098 // Declares the flags.
00099
00100 // This flag temporary enables the disabled tests.
00101 GTEST_DECLARE_bool_(also_run_disabled_tests);
00102
00103 // This flag brings the debugger on an assertion failure.
00104 GTEST_DECLARE_bool_(break_on_failure);
00105
00106 // This flag controls whether Google Test catches all test-thrown exceptions
00107 // and logs them as failures.
00108 GTEST_DECLARE_bool_(catch_exceptions);
00109
00110 // This flag enables using colors in terminal output. Available values are
00111 // "yes" to enable colors, "no" (disable colors), or "auto" (the default)
00112 // to let Google Test decide.
00113 GTEST_DECLARE_string_(color);
00114
00115 // This flag sets up the filter to select by name using a glob pattern
00116 // the tests to run. If the filter is not given all tests are executed.
00117 GTEST_DECLARE_string_(filter);
00118
00119 // This flag controls whether Google Test installs a signal handler that dumps
00120 // debugging information when fatal signals are raised.
00121 GTEST_DECLARE_bool_(install_failure_signal_handler);
00122
00123 // This flag causes the Google Test to list tests. None of the tests listed
00124 // are actually run if the flag is provided.
00125 GTEST_DECLARE_bool_(list_tests);
00126
00127 // This flag controls whether Google Test emits a detailed XML report to a file
00128 // in addition to its normal textual output.
00129 GTEST_DECLARE_string_(output);
00130
00131 // This flags control whether Google Test prints the elapsed time for each
00132 // test.
00133 GTEST_DECLARE_bool_(print_time);
00134
00135 // This flags control whether Google Test prints UTF8 characters as text.
00136 GTEST_DECLARE_bool_(print_utf8);
00137
00138 // This flag specifies the random number seed.
00139 GTEST_DECLARE_int32_(random_seed);
00140
00141 // This flag sets how many times the tests are repeated. The default value
00142 // is 1. If the value is -1 the tests are repeating forever.
00143 GTEST_DECLARE_int32_(repeat);
00144
00145 // This flag controls whether Google Test includes Google Test internal
```

```
00146 // stack frames in failure stack traces.
00147 #include <gtest/gtest.h>
00148
00149 // When this flag is specified, tests' order is randomized on every iteration.
00150 #include <gtest/gtest.h>
00151
00152 // This flag specifies the maximum number of stack frames to be
00153 // printed in a failure message.
00154 #include <gtest/gtest.h>
00155
00156 // When this flag is specified, a failed assertion will throw an
00157 // exception if exceptions are enabled, or exit the program with a
00158 // non-zero code otherwise. For use with an external test framework.
00159 #include <gtest/gtest.h>
00160
00161 // When this flag is set with a "host:port" string, on supported
00162 // platforms test results are streamed to the specified port on
00163 // the specified host machine.
00164 #include <gtest/gtest.h>
00165
00166 #if GTEST_USE_OWN_FLAGFILE_FLAG_
00167 #include <gtest/gtest.h>
00168 #endif // GTEST_USE_OWN_FLAGFILE_FLAG_
00169
00170 // The upper limit for valid stack trace depths.
00171 const int kMaxStackTraceDepth = 100;
00172
00173 namespace internal {
00174
00175 class AssertHelper;
00176 class DefaultGlobalTestPartResultReporter;
00177 class ExecDeathTest;
00178 class NoExecDeathTest;
00179 class FinalSuccessChecker;
00180 class GTestFlagSaver;
00181 class StreamingListenerTest;
00182 class TestResultAccessor;
00183 class TestEventListenersAccessor;
00184 class TestEventRepeater;
00185 class UnitTestRecordPropertyTestHelper;
00186 class WindowsDeathTest;
00187 class FuchsiaDeathTest;
00188 class UnitTestImpl* GetUnitTestImpl();
00189 void ReportFailureInUnknownLocation(TestPartResult::Type result_type,
00190                                     const std::string& message);
00191
00192 } // namespace internal
00193
00194 // The friend relationship of some of these classes is cyclic.
00195 // If we don't forward declare them the compiler might confuse the classes
00196 // in friendship clauses with same named classes on the scope.
00197 class Test;
00198 class TestCase;
00199 class TestInfo;
00200 class UnitTest;
00201
00202 // A class for indicating whether an assertion was successful. When
00203 // the assertion wasn't successful, the AssertionResult object
00204 // remembers a non-empty message that describes how it failed.
00205 //
00206 // To create an instance of this class, use one of the factory functions
00207 // (AssertionSuccess() and AssertionFailure()).
00208 //
00209 // This class is useful for two purposes:
00210 //   1. Defining predicate functions to be used with Boolean test assertions
00211 //      EXPECT_TRUE/EXPECT_FALSE and their ASSERT_ counterparts
00212 //   2. Defining predicate-format functions to be
00213 //      used with predicate assertions (ASSERT_PRED_FORMAT*, etc).
00214 //
00215 // For example, if you define IsEven predicate:
00216 //
00217 //   testing::AssertionResult IsEven(int n) {
00218 //     if ((n % 2) == 0)
00219 //       return testing::AssertionSuccess();
00220 //     else
00221 //       return testing::AssertionFailure() << n << " is odd";
00222 //   }
00223 //
00224 // Then the failed expectation EXPECT_TRUE(IsEven(Fib(5)))
00225 // will print the message
00226 //
00227 //   Value of: IsEven(Fib(5))
00228 //   Actual: false (5 is odd)
00229 //   Expected: true
00230 //
00231 // instead of a more opaque
00232 //
```

```

00233 //      Value of: IsEven(Fib(5))
00234 //      Actual: false
00235 //      Expected: true
00236 //
00237 // in case IsEven is a simple Boolean predicate.
00238 //
00239 // If you expect your predicate to be reused and want to support informative
00240 // messages in EXPECT_FALSE and ASSERT_FALSE (negative assertions show up
00241 // about half as often as positive ones in our tests), supply messages for
00242 // both success and failure cases:
00243 //
00244 //     testing::AssertionResult IsEven(int n) {
00245 //         if ((n % 2) == 0)
00246 //             return testing::AssertionSuccess() << n << " is even";
00247 //         else
00248 //             return testing::AssertionFailure() << n << " is odd";
00249 //     }
00250 //
00251 // Then a statement EXPECT_FALSE(IsEven(Fib(6))) will print
00252 //
00253 //      Value of: IsEven(Fib(6))
00254 //      Actual: true (8 is even)
00255 //      Expected: false
00256 //
00257 // NB: Predicates that support negative Boolean assertions have reduced
00258 // performance in positive ones so be careful not to use them in tests
00259 // that have lots (tens of thousands) of positive Boolean assertions.
00260 //
00261 // To use this class with EXPECT_PRED_FORMAT assertions such as:
00262 //
00263 //     // Verifies that Foo() returns an even number.
00264 //     EXPECT_PRED_FORMAT1(IsEven, Foo());
00265 //
00266 // you need to define:
00267 //
00268 //     testing::AssertionResult IsEven(const char* expr, int n) {
00269 //         if ((n % 2) == 0)
00270 //             return testing::AssertionSuccess();
00271 //         else
00272 //             return testing::AssertionFailure()
00273 //                 << "Expected: " << expr << " is even\n Actual: it's " << n;
00274 //     }
00275 //
00276 // If Foo() returns 5, you will see the following message:
00277 //
00278 //     Expected: Foo() is even
00279 //     Actual: it's 5
00280 //
00281 class GTEST_API_ AssertionResult {
00282 public:
00283     // Copy constructor.
00284     // Used in EXPECT_TRUE/EXPECT_FALSE(assertion_result).
00285     AssertionResult(const AssertionResult& other);
00286
00287 #if defined(_MSC_VER) && _MSC_VER < 1910
00288     GTEST_DISABLE_MSC_WARNINGS_PUSH_(4800 /* forcing value to bool */)
00289 #endif
00290
00291     // Used in the EXPECT_TRUE/EXPECT_FALSE(bool_expression).
00292     //
00293     // T must be contextually convertible to bool.
00294     //
00295     // The second parameter prevents this overload from being considered if
00296     // the argument is implicitly convertible to AssertionResult. In that case
00297     // we want AssertionResult's copy constructor to be used.
00298     template <typename T>
00299     explicit AssertionResult(
00300         const T& success,
00301         typename internal::EnableIf<
00302             !internal::ImplicitlyConvertible<T, AssertionResult>::value>::type*
00303             /*enabler*/ = NULL)
00304         : success_(success) {}
00305
00306 #if defined(_MSC_VER) && _MSC_VER < 1910
00307     GTEST_DISABLE_MSC_WARNINGS_POP_()
00308 #endif
00309
00310     // Assignment operator.
00311     AssertionResult& operator=(AssertionResult other) {
00312         swap(other);
00313         return *this;
00314     }
00315
00316     // Returns true iff the assertion succeeded.
00317     operator bool() const { return success_; } // NOLINT
00318
00319     // Returns the assertion's negation. Used with EXPECT/ASSERT_FALSE.

```

```
00320     AssertionResult operator!() const;
00321
00322     // Returns the text streamed into this AssertionResult. Test assertions
00323     // use it when they fail (i.e., the predicate's outcome doesn't match the
00324     // assertion's expectation). When nothing has been streamed into the
00325     // object, returns an empty string.
00326     const char* message() const {
00327         return message_.get() != NULL ? message_->c_str() : "";
00328     }
00329     // FIXME: Remove this after making sure no clients use it.
00330     // Deprecated; please use message() instead.
00331     const char* failure_message() const { return message(); }
00332
00333     // Streams a custom failure message into this object.
00334     template <typename T> AssertionResult& operator<<(const T& value) {
00335         AppendMessage(Message() << value);
00336         return *this;
00337     }
00338
00339     // Allows streaming basic output manipulators such as endl or flush into
00340     // this object.
00341     AssertionResult& operator<<(
00342         ::std::ostream& (*basic_manipulator)(::std::ostream& stream)) {
00343         AppendMessage(Message() << basic_manipulator);
00344         return *this;
00345     }
00346
00347 private:
00348     // Appends the contents of message to message_.
00349     void AppendMessage(const Message& a_message) {
00350         if (message_.get() == NULL)
00351             message_.reset(new ::std::string);
00352         message_->append(a_message.GetString().c_str());
00353     }
00354
00355     // Swap the contents of this AssertionResult with other.
00356     void swap(AssertionResult& other);
00357
00358     // Stores result of the assertion predicate.
00359     bool success_;
00360
00361     // Stores the message describing the condition in case the expectation
00362     // construct is not satisfied with the predicate's outcome.
00363     // Referenced via a pointer to avoid taking too much stack frame space
00364     // with test assertions.
00365     internal::scoped_ptr< ::std::string> message_;
00366
00367     // Makes a successful assertion result.
00368     GTEST_API_ AssertionResult AssertionSuccess();
00369
00370     // Makes a failed assertion result.
00371     GTEST_API_ AssertionResult AssertionFailure();
00372
00373     // Makes a failed assertion result with the given failure message.
00374     // Deprecated; use AssertionFailure() << msg.
00375     GTEST_API_ AssertionResult AssertionFailure(const Message& msg);
00376
00377 } // namespace testing
00378
00379 // Includes the auto-generated header that implements a family of generic
00380 // predicate assertion macros. This include comes late because it relies on
00381 // APIs declared above.
00382 #include "gtest/gtest_pred_impl.h"
00383
00384 namespace testing {
00385
00386     // The abstract class that all tests inherit from.
00387
00388     // In Google Test, a unit test program contains one or many TestCases, and
00389     // each TestCase contains one or many Tests.
00390
00391     // When you define a test using the TEST macro, you don't need to
00392     // explicitly derive from Test - the TEST macro automatically does
00393     // this for you.
00394
00395     // The only time you derive from Test is when defining a test fixture
00396     // to be used in a TEST_F. For example:
00397
00398     class FooTest : public testing::Test {
00399         // protected:
00400         void SetUp() override { ... }
00401         void TearDown() override { ... }
00402         ...
00403     };
00404
00405     // TEST_F(FooTest, Bar) { ... }
00406     // TEST_F(FooTest, Baz) { ... }
```

```

00407 //
00408 // Test is not copyable.
00409 class GTEST_API_ Test {
00410 public:
00411 friend class TestInfo;
00412
00413 // Defines types for pointers to functions that set up and tear down
00414 // a test case.
00415 typedef internal::SetUpTestCaseFunc SetUpTestCaseFunc;
00416 typedef internal::TearDownTestCaseFunc TearDownTestCaseFunc;
00417
00418 // The d'tor is virtual as we intend to inherit from Test.
00419 virtual ~Test();
00420
00421 // Sets up the stuff shared by all tests in this test case.
00422 //
00423 // Google Test will call Foo::SetUpTestCase() before running the first
00424 // test in test case Foo. Hence a sub-class can define its own
00425 // SetUpTestCase() method to shadow the one defined in the super
00426 // class.
00427 static void SetUpTestCase() {}
00428
00429 // Tears down the stuff shared by all tests in this test case.
00430 //
00431 // Google Test will call Foo::TearDownTestCase() after running the last
00432 // test in test case Foo. Hence a sub-class can define its own
00433 // TearDownTestCase() method to shadow the one defined in the super
00434 // class.
00435 static void TearDownTestCase() {}
00436
00437 // Returns true iff the current test has a fatal failure.
00438 static bool HasFatalFailure();
00439
00440 // Returns true iff the current test has a non-fatal failure.
00441 static bool HasNonfatalFailure();
00442
00443 // Returns true iff the current test has a (either fatal or
00444 // non-fatal) failure.
00445 static bool HasFailure() { return HasFatalFailure() || HasNonfatalFailure(); }
00446
00447 // Logs a property for the current test, test case, or for the entire
00448 // invocation of the test program when used outside of the context of a
00449 // test case. Only the last value for a given key is remembered. These
00450 // are public static so they can be called from utility functions that are
00451 // not members of the test fixture. Calls to RecordProperty made during
00452 // lifespan of the test (from the moment its constructor starts to the
00453 // moment its destructor finishes) will be output in XML as attributes of
00454 // the <testcase> element. Properties recorded from fixture's
00455 // SetUpTestCase or TearDownTestCase are logged as attributes of the
00456 // corresponding <testsuite> element. Calls to RecordProperty made in the
00457 // global context (before or after invocation of RUN_ALL_TESTS and from
00458 // SetUp/TearDown method of Environment objects registered with Google
00459 // Test) will be output as attributes of the <testsuites> element.
00460 static void RecordProperty(const std::string& key, const std::string& value);
00461 static void RecordProperty(const std::string& key, int value);
00462
00463 protected:
00464 // Creates a Test object.
00465 Test();
00466
00467 // Sets up the test fixture.
00468 virtual void SetUp();
00469
00470 // Tears down the test fixture.
00471 virtual void TearDown();
00472
00473 private:
00474 // Returns true iff the current test has the same fixture class as
00475 // the first test in the current test case.
00476 static bool HasSameFixtureClass();
00477
00478 // Runs the test after the test fixture has been set up.
00479 //
00480 // A sub-class must implement this to define the test logic.
00481 //
00482 // DO NOT OVERRIDE THIS FUNCTION DIRECTLY IN A USER PROGRAM.
00483 // Instead, use the TEST or TEST_F macro.
00484 virtual void TestBody() = 0;
00485
00486 // Sets up, executes, and tears down the test.
00487 void Run();
00488
00489 // Deletes self. We deliberately pick an unusual name for this
00490 // internal method to avoid clashing with names used in user TESTs.
00491 void DeleteSelf_() { delete this; }
00492
00493 const internal::scoped_ptr< GTEST_FLAG_SAVER_ > gtest_flag_saver_;

```

```
00494
00495 // Often a user misspells SetUp() as Setup() and spends a long time
00496 // wondering why it is never called by Google Test. The declaration of
00497 // the following method is solely for catching such an error at
00498 // compile time:
00499 //
00500 // - The return type is deliberately chosen to be not void, so it
00501 // will be a conflict if void Setup() is declared in the user's
00502 // test fixture.
00503 //
00504 // - This method is private, so it will be another compiler error
00505 // if the method is called from the user's test fixture.
00506 //
00507 // DO NOT OVERRIDE THIS FUNCTION.
00508 //
00509 // If you see an error about overriding the following function or
00510 // about it being private, you have mis-spelled SetUp() as Setup().
00511 struct Setup_should_be_spelled_SetUp {};
00512 virtual Setup_should_be_spelled_SetUp* Setup() { return NULL; }
00513
00514 // We disallow copying Tests.
00515 GTEST_DISALLOW_COPY_AND_ASSIGN_(Test);
00516 };
00517
00518 typedef internal::TimeInMillis TimeInMillis;
00519
00520 // A copyable object representing a user specified test property which can be
00521 // output as a key/value string pair.
00522 //
00523 // Don't inherit from TestProperty as its destructor is not virtual.
00524 class TestProperty {
00525 public:
00526 // C'tor. TestProperty does NOT have a default constructor.
00527 // Always use this constructor (with parameters) to create a
00528 // TestProperty object.
00529 TestProperty(const std::string& a_key, const std::string& a_value) :
00530     key_(a_key), value_(a_value)
00531 }
00532
00533 // Gets the user supplied key.
00534 const char* key() const {
00535     return key_.c_str();
00536 }
00537
00538 // Gets the user supplied value.
00539 const char* value() const {
00540     return value_.c_str();
00541 }
00542
00543 // Sets a new value, overriding the one supplied in the constructor.
00544 void SetValue(const std::string& new_value) {
00545     value_ = new_value;
00546 }
00547
00548 private:
00549 // The key supplied by the user.
00550 std::string key_;
00551 // The value supplied by the user.
00552 std::string value_;
00553 };
00554
00555 // The result of a single Test. This includes a list of
00556 // TestPartResults, a list of TestProperties, a count of how many
00557 // death tests there are in the Test, and how much time it took to run
00558 // the Test.
00559 //
00560 // TestResult is not copyable.
00561 class GTEST_API_ TestResult {
00562 public:
00563 // Creates an empty TestResult.
00564 TestResult();
00565
00566 // D'tor. Do not inherit from TestResult.
00567 ~TestResult();
00568
00569 // Gets the number of all test parts. This is the sum of the number
00570 // of successful test parts and the number of failed test parts.
00571 int total_part_count() const;
00572
00573 // Returns the number of the test properties.
00574 int test_property_count() const;
00575
00576 // Returns true iff the test passed (i.e. no test part failed).
00577 bool Passed() const { return !Failed(); }
00578
00579 // Returns true iff the test failed.
00580 bool Failed() const;
```

```
00581 // Returns true iff the test fatally failed.
00582 bool HasFatalFailure() const;
00583
00584 // Returns true iff the test has a non-fatal failure.
00585 bool HasNonfatalFailure() const;
00586
00587 // Returns the elapsed time, in milliseconds.
00588 TimeInMillis elapsed_time() const { return elapsed_time_; }
00589
00590 // Returns the i-th test part result among all the results. i can range from 0
00591 // to total_part_count() - 1. If i is not in that range, aborts the program.
00592 const TestPartResult& GetTestPartResult(int i) const;
00593
00594 // Returns the i-th test property. i can range from 0 to
00595 // test_property_count() - 1. If i is not in that range, aborts the
00596 // program.
00597 const TestProperty& GetTestProperty(int i) const;
00598
00599 private:
00600     friend class TestInfo;
00601     friend class TestCase;
00602     friend class UnitTest;
00603     friend class internal::DefaultGlobalTestPartResultReporter;
00604     friend class internal::ExecDeathTest;
00605     friend class internal::TestResultAccessor;
00606     friend class internal::UnitTestImpl;
00607     friend class internal::WindowsDeathTest;
00608     friend class internal::FuchsiaDeathTest;
00609
00610 // Gets the vector of TestPartResults.
00611 const std::vector<TestPartResult>& test_part_results() const {
00612     return test_part_results_;
00613 }
00614
00615 // Gets the vector of TestProperties.
00616 const std::vector<TestProperty>& test_properties() const {
00617     return test_properties_;
00618 }
00619
00620 // Sets the elapsed time.
00621 void set_elapsed_time(TimeInMillis elapsed) { elapsed_time_ = elapsed; }
00622
00623 // Adds a test property to the list. The property is validated and may add
00624 // a non-fatal failure if invalid (e.g., if it conflicts with reserved
00625 // key names). If a property is already recorded for the same key, the
00626 // value will be updated, rather than storing multiple values for the same
00627 // key. xml_element specifies the element for which the property is being
00628 // recorded and is used for validation.
00629 void RecordProperty(const std::string& xml_element,
00630                     const TestProperty& test_property);
00631
00632 // Adds a failure if the key is a reserved attribute of Google Test
00633 // testcase tags. Returns true if the property is valid.
00634 // FIXME: Validate attribute names are legal and human readable.
00635 static bool ValidateTestProperty(const std::string& xml_element,
00636                                 const TestProperty& test_property);
00637
00638 // Adds a test part result to the list.
00639 void AddTestPartResult(const TestPartResult& test_part_result);
00640
00641 // Returns the death test count.
00642 int death_test_count() const { return death_test_count_; }
00643
00644 // Increments the death test count, returning the new count.
00645 int increment_death_test_count() { return ++death_test_count_; }
00646
00647 // Clears the test part results.
00648 void ClearTestPartResults();
00649
00650 // Clears the object.
00651 void Clear();
00652
00653 // Protects mutable state of the property vector and of owned
00654 // properties, whose values may be updated.
00655 internal::Mutex test_properties_mutex_;
00656
00657 // The vector of TestPartResults
00658 std::vector<TestPartResult> test_part_results_;
00659 // The vector of TestProperties
00660 std::vector<TestProperty> test_properties_;
00661 // Running count of death tests.
00662 int death_test_count_;
00663 // The elapsed time, in milliseconds.
00664 TimeInMillis elapsed_time_;
00665
00666 // We disallow copying TestResult.
```

```
00668     GTEST_DISALLOW_COPY_AND_ASSIGN_(TestResult);
00669 }; // class TestResult
0070
0071 // A TestInfo object stores the following information about a test:
0072 //
0073 //   Test case name
0074 //   Test name
0075 //   Whether the test should be run
0076 //   A function pointer that creates the test object when invoked
0077 //   Test result
0078 //
0079 // The constructor of TestInfo registers itself with the UnitTest
0080 // singleton such that the RUN_ALL_TESTS() macro knows which tests to
0081 // run.
0082 class GTEST_API_ TestInfo {
0083 public:
0084     // Destroys a TestInfo object. This function is not virtual, so
0085     // don't inherit from TestInfo.
0086     ~TestInfo();
0087
0088     // Returns the test case name.
0089     const char* test_case_name() const { return test_case_name_.c_str(); }
0090
0091     // Returns the test name.
0092     const char* name() const { return name_.c_str(); }
0093
0094     // Returns the name of the parameter type, or NULL if this is not a typed
0095     // or a type-parameterized test.
0096     const char* type_param() const {
0097         if (type_param_.get() != NULL)
0098             return type_param_->c_str();
0099         return NULL;
0100     }
0101
0102     // Returns the text representation of the value parameter, or NULL if this
0103     // is not a value-parameterized test.
0104     const char* value_param() const {
0105         if (value_param_.get() != NULL)
0106             return value_param_->c_str();
0107         return NULL;
0108     }
0109
0110     // Returns the file name where this test is defined.
0111     const char* file() const { return location_.file.c_str(); }
0112
0113     // Returns the line where this test is defined.
0114     int line() const { return location_.line; }
0115
0116     // Return true if this test should not be run because it's in another shard.
0117     bool is_in_another_shard() const { return is_in_another_shard_; }
0118
0119     // Returns true if this test should run, that is if the test is not
0120     // disabled (or it is disabled but the also_run_disabled_tests flag has
0121     // been specified) and its full name matches the user-specified filter.
0122     //
0123     // Google Test allows the user to filter the tests by their full names.
0124     // The full name of a test Bar in test case Foo is defined as
0125     // "Foo.Bar". Only the tests that match the filter will run.
0126     //
0127     // A filter is a colon-separated list of glob (not regex) patterns,
0128     // optionally followed by a '-' and a colon-separated list of
0129     // negative patterns (tests to exclude). A test is run if it
0130     // matches one of the positive patterns and does not match any of
0131     // the negative patterns.
0132     //
0133     // For example, *A*:Foo.* is a filter that matches any string that
0134     // contains the character 'A' or starts with "Foo.".
0135     bool should_run() const { return should_run_; }
0136
0137     // Returns true iff this test will appear in the XML report.
0138     bool is_reportable() const {
0139         // The XML report includes tests matching the filter, excluding those
0140         // run in other shards.
0141         return matches_filter_ && !is_in_another_shard_;
0142     }
0143
0144     // Returns the result of the test.
0145     const TestResult* result() const { return &result_; }
0146
0147 private:
0148 #if GTEST_HAS_DEATH_TEST
0149     friend class internal::DefaultDeathTestFactory;
0150 #endif // GTEST_HAS_DEATH_TEST
0151     friend class Test;
0152     friend class TestCase;
0153     friend class internal::UnitTestImpl;
0154     friend class internal::StreamingListenerTest;
```

```

00755     friend TestInfo* internal::MakeAndRegisterTestInfo(
00756         const char* test_case_name,
00757         const char* name,
00758         const char* type_param,
00759         const char* value_param,
00760         internal::CodeLocation code_location,
00761         internal::TypeId fixture_class_id,
00762         Test::SetUpTestCaseFunc set_up_tc,
00763         Test::TearDownTestCaseFunc tear_down_tc,
00764         internal::TestFactoryBase* factory);
00765
00766     // Constructs a TestInfo object. The newly constructed instance assumes
00767     // ownership of the factory object.
00768     TestInfo(const std::string& test_case_name,
00769             const std::string& name,
00770             const char* a_type_param,    // NULL if not a type-parameterized test
00771             const char* a_value_param,  // NULL if not a value-parameterized test
00772             internal::CodeLocation a_code_location,
00773             internal::TypeId fixture_class_id,
00774             internal::TestFactoryBase* factory);
00775
00776     // Increments the number of death tests encountered in this test so
00777     // far.
00778     int increment_death_test_count() {
00779         return result_.increment_death_test_count();
00780     }
00781
00782     // Creates the test object, runs it, records its result, and then
00783     // deletes it.
00784     void Run();
00785
00786     static void ClearTestResult(TestInfo* test_info) {
00787         test_info->result_.Clear();
00788     }
00789
00790     // These fields are immutable properties of the test.
00791     const std::string test_case_name_;      // Test case name
00792     const std::string name_;                // Test name
00793     // Name of the parameter type, or NULL if this is not a typed or a
00794     // type-parameterized test.
00795     const internal::scoped_ptr<const std::string> type_param_;
00796     // Text representation of the value parameter, or NULL if this is not a
00797     // value-parameterized test.
00798     const internal::scoped_ptr<const std::string> value_param_;
00799     internal::CodeLocation location_;
00800     const internal::TypeId fixture_class_id_; // ID of the test fixture class
00801     bool should_run_;                      // True iff this test should run
00802     bool is_disabled_;                     // True iff this test is disabled
00803     bool matches_filter_;                  // True if this test matches the
00804                                         // user-specified filter.
00805     bool is_in_another_shard_;            // Will be run in another shard.
00806     internal::TestFactoryBase* const factory_; // The factory that creates
00807                                         // the test object
00808
00809     // This field is mutable and needs to be reset before running the
00810     // test for the second time.
00811     TestResult result_;
00812
00813     GTEST_DISALLOW_COPY_AND_ASSIGN_(TestInfo);
00814 };
00815
00816 // A test case, which consists of a vector of TestInfos.
00817 //
00818 // TestCase is not copyable.
00819 class GTEST_API_ TestCase {
00820 public:
00821     // Creates a TestCase with the given name.
00822     //
00823     // TestCase does NOT have a default constructor. Always use this
00824     // constructor to create a TestCase object.
00825     //
00826     // Arguments:
00827     //
00828     //   name:          name of the test case
00829     //   a_type_param:  the name of the test's type parameter, or NULL if
00830     //                 this is not a type-parameterized test.
00831     //   set_up_tc:     pointer to the function that sets up the test case
00832     //   tear_down_tc:  pointer to the function that tears down the test case
00833     TestCase(const char* name, const char* a_type_param,
00834              Test::SetUpTestCaseFunc set_up_tc,
00835              Test::TearDownTestCaseFunc tear_down_tc);
00836
00837     // Destructor of TestCase.
00838     virtual ~TestCase();
00839
00840     // Gets the name of the TestCase.
00841     const char* name() const { return name_.c_str(); }

```

```
00842 // Returns the name of the parameter type, or NULL if this is not a
00843 // type-parameterized test case.
00844 const char* type_param() const {
00845     if (type_param_.get() != NULL)
00846         return type_param_->c_str();
00847     return NULL;
00848 }
00849
00850 // Returns true if any test in this test case should run.
00851 bool should_run() const { return should_run_; }
00852
00853 // Gets the number of successful tests in this test case.
00854 int successful_test_count() const;
00855
00856 // Gets the number of failed tests in this test case.
00857 int failed_test_count() const;
00858
00859 // Gets the number of disabled tests that will be reported in the XML report.
00860 int reportable_disabled_test_count() const;
00861
00862 // Gets the number of disabled tests in this test case.
00863 int disabled_test_count() const;
00864
00865 // Gets the number of tests to be printed in the XML report.
00866 int reportable_test_count() const;
00867
00868 // Get the number of tests in this test case that should run.
00869 int test_to_run_count() const;
00870
00871 // Gets the number of all tests in this test case.
00872 int total_test_count() const;
00873
00874 // Returns true iff the test case passed.
00875 bool Passed() const { return !Failed(); }
00876
00877 // Returns true iff the test case failed.
00878 bool Failed() const { return failed_test_count() > 0; }
00879
00880 // Returns the elapsed time, in milliseconds.
00881 TimeInMillis elapsed_time() const { return elapsed_time_; }
00882
00883 // Returns the i-th test among all the tests. i can range from 0 to
00884 // total_test_count() - 1. If i is not in that range, returns NULL.
00885 const TestInfo* GetTestInfo(int i) const;
00886
00887 // Returns the TestResult that holds test properties recorded during
00888 // execution of SetUpTestCase and TearDownTestCase.
00889 const TestResult& ad_hoc_test_result() const { return ad_hoc_test_result_; }
00890
00891 private:
00892     friend class Test;
00893     friend class internal::UnitTestImpl;
00894
00895 // Gets the (mutable) vector of TestInfos in this TestCase.
00896 std::vector<TestInfo*>& test_info_list() { return test_info_list_; }
00897
00898 // Gets the (immutable) vector of TestInfos in this TestCase.
00899 const std::vector<TestInfo*>& test_info_list() const {
00900     return test_info_list_;
00901 }
00902
00903 // Returns the i-th test among all the tests. i can range from 0 to
00904 // total_test_count() - 1. If i is not in that range, returns NULL.
00905 const TestInfo* GetMutableTestInfo(int i);
00906
00907 // Sets the should_run member.
00908 void set_should_run(bool should) { should_run_ = should; }
00909
00910 // Adds a TestInfo to this test case. Will delete the TestInfo upon
00911 // destruction of the TestCase object.
00912 void AddTestInfo(TestInfo * test_info);
00913
00914 // Clears the results of all tests in this test case.
00915 void ClearResult();
00916
00917 // Clears the results of all tests in the given test case.
00918 static void ClearTestCaseResult(TestCase* test_case) {
00919     test_case->ClearResult();
00920 }
00921
00922 // Runs every test in this TestCase.
00923 void Run();
00924
00925 // Runs SetUpTestCase() for this TestCase. This wrapper is needed
00926 // for catching exceptions thrown from SetUpTestCase().
00927 void RunSetUpTestCase() { (*set_up_tc_)(); }
```

```
00929
00930 // Runs TearDownTestCase() for this TestCase. This wrapper is
00931 // needed for catching exceptions thrown from TearDownTestCase().
00932 void RunTearDownTestCase() { (*tear_down_tc_)(); }
00933
00934 // Returns true iff test passed.
00935 static bool TestPassed(const TestInfo* test_info) {
00936     return test_info->should_run() && test_info->result()->Passed();
00937 }
00938
00939 // Returns true iff test failed.
00940 static bool TestFailed(const TestInfo* test_info) {
00941     return test_info->should_run() && test_info->result()->Failed();
00942 }
00943
00944 // Returns true iff the test is disabled and will be reported in the XML
00945 // report.
00946 static bool TestReportableDisabled(const TestInfo* test_info) {
00947     return test_info->is_reportable() && test_info->is_disabled_;
00948 }
00949
00950 // Returns true iff test is disabled.
00951 static bool TestDisabled(const TestInfo* test_info) {
00952     return test_info->is_disabled_;
00953 }
00954
00955 // Returns true iff this test will appear in the XML report.
00956 static bool TestReportable(const TestInfo* test_info) {
00957     return test_info->is_reportable();
00958 }
00959
00960 // Returns true if the given test should run.
00961 static bool ShouldRunTest(const TestInfo* test_info) {
00962     return test_info->should_run();
00963 }
00964
00965 // Shuffles the tests in this test case.
00966 void ShuffleTests(internal::Random* random);
00967
00968 // Restores the test order to before the first shuffle.
00969 void UnshuffleTests();
00970
00971 // Name of the test case.
00972 std::string name_;
00973 // Name of the parameter type, or NULL if this is not a typed or a
00974 // type-parameterized test.
00975 const internal::scoped_ptr<const std::string> type_param_;
00976 // The vector of TestInfos in their original order. It owns the
00977 // elements in the vector.
00978 std::vector<TestInfo*> test_info_list_;
00979 // Provides a level of indirection for the test list to allow easy
00980 // shuffling and restoring the test order. The i-th element in this
00981 // vector is the index of the i-th test in the shuffled test list.
00982 std::vector<int> test_indices_;
00983 // Pointer to the function that sets up the test case.
00984 Test::SetUpTestCaseFunc set_up_tc_;
00985 // Pointer to the function that tears down the test case.
00986 Test::TearDownTestCaseFunc tear_down_tc_;
00987 // True iff any test in this test case should run.
00988 bool should_run_;
00989 // Elapsed time, in milliseconds.
00990 TimeInMillis elapsed_time_;
00991 // Holds test properties recorded during execution of SetUpTestCase and
00992 // TearDownTestCase.
00993 TestResult ad_hoc_test_result_;
00994
00995 // We disallow copying TestCases.
00996 GTEST_DISALLOW_COPY_AND_ASSIGN_(TestCase);
00997 };
00998
00999 // An Environment object is capable of setting up and tearing down an
01000 // environment. You should subclass this to define your own
01001 // environment(s).
01002 //
01003 // An Environment object does the set-up and tear-down in virtual
01004 // methods SetUp() and TearDown() instead of the constructor and the
01005 // destructor, as:
01006 //
01007 // 1. You cannot safely throw from a destructor. This is a problem
01008 //      as in some cases Google Test is used where exceptions are enabled, and
01009 //      we may want to implement ASSERT_* using exceptions where they are
01010 //      available.
01011 // 2. You cannot use ASSERT_* directly in a constructor or
01012 //      destructor.
01013 class Environment {
01014 public:
01015     // The d'tor is virtual as we need to subclass Environment.
```

```
01016     virtual ~Environment() {}
01017
01018     // Override this to define how to set up the environment.
01019     virtual void SetUp() {}
01020
01021     // Override this to define how to tear down the environment.
01022     virtual void TearDown() {}
01023
01024     // If you see an error about overriding the following function or
01025     // about it being private, you have mis-spelled SetUp() as Setup().
01026     struct Setup_should_be_spelled_SetUp {};
01027     virtual Setup_should_be_spelled_SetUp* Setup() { return NULL; }
01028 };
01029
01030 #if GTEST_HAS_EXCEPTIONS
01031
01032 // Exception which can be thrown from TestEventListener::OnTestPartResult.
01033 class GTEST_API_ AssertionException
01034     : public internal::GoogleTestFailureException {
01035
01036     public:
01037         explicit AssertionException(const TestPartResult& result)
01038         : GoogleTestFailureException(result) {}
01039
01040 #endif // GTEST_HAS_EXCEPTIONS
01041
01042 // The interface for tracing execution of tests. The methods are organized in
01043 // the order the corresponding events are fired.
01044 class TestEventListener {
01045
01046     public:
01047         virtual ~TestEventListener() {}
01048
01049         // Fired before any test activity starts.
01050         virtual void OnTestProgramStart(const UnitTest& unit_test) = 0;
01051
01052         // Fired before each iteration of tests starts. There may be more than
01053         // one iteration if GTEST_FLAG(repeat) is set. iteration is the iteration
01054         // index, starting from 0.
01055         virtual void OnTestIterationStart(const UnitTest& unit_test,
01056                                         int iteration) = 0;
01057
01058         // Fired before environment set-up for each iteration of tests starts.
01059         virtual void OnEnvironmentsSetUpStart(const UnitTest& unit_test) = 0;
01060
01061         // Fired after environment set-up for each iteration of tests ends.
01062         virtual void OnEnvironmentsSetUpEnd(const UnitTest& unit_test) = 0;
01063
01064         // Fired before the test case starts.
01065         virtual void OnTestCaseStart(const TestCase& test_case) = 0;
01066
01067         // Fired before the test starts.
01068         virtual void OnTestStart(const TestInfo& test_info) = 0;
01069
01070         // Fired after a failed assertion or a SUCCEED() invocation.
01071         // If you want to throw an exception from this function to skip to the next
01072         // TEST, it must be AssertionException defined above, or inherited from it.
01073         virtual void OnTestPartResult(const TestPartResult& test_part_result) = 0;
01074
01075         // Fired after the test ends.
01076         virtual void OnTestEnd(const TestInfo& test_info) = 0;
01077
01078         // Fired after the test case ends.
01079         virtual void OnTestCaseEnd(const TestCase& test_case) = 0;
01080
01081         // Fired before environment tear-down for each iteration of tests starts.
01082         virtual void OnEnvironmentsTearDownStart(const UnitTest& unit_test) = 0;
01083
01084         // Fired after environment tear-down for each iteration of tests ends.
01085         virtual void OnEnvironmentsTearDownEnd(const UnitTest& unit_test) = 0;
01086
01087         // Fired after each iteration of tests finishes.
01088         virtual void OnTestIterationEnd(const UnitTest& unit_test,
01089                                         int iteration) = 0;
01090
01091         // Fired after all test activities have ended.
01092         virtual void OnTestProgramEnd(const UnitTest& unit_test) = 0;
01093
01094 // The convenience class for users who need to override just one or two
01095 // methods and are not concerned that a possible change to a signature of
01096 // the methods they override will not be caught during the build. For
01097 // comments about each method please see the definition of TestEventListener
01098 // above.
01099 class EmptyTestEventListener : public TestEventListener {
01100
01101     public:
01102         virtual void OnTestProgramStart(const UnitTest& /*unit_test*/) {}
01103         virtual void OnTestIterationStart(const UnitTest& /*unit_test*/,
```

```

01103             int /*iteration*/) {}
01104     virtual void OnEnvironmentsSetUpStart(const UnitTest /*unit_test*/) {}
01105     virtual void OnEnvironmentsSetUpEnd(const UnitTest& /*unit_test*/) {}
01106     virtual void OnTestCaseStart(const TestCase& /*test_case*/) {}
01107     virtual void OnTestStart(const TestInfo& /*test_info*/) {}
01108     virtual void OnTestPartResult(const TestPartResult& /*test_part_result*/) {}
01109     virtual void OnTestEnd(const TestInfo& /*test_info*/) {}
01110     virtual void OnTestCaseEnd(const TestCase& /*test_case*/) {}
01111     virtual void OnEnvironmentsTearDownStart(const UnitTest& /*unit_test*/) {}
01112     virtual void OnEnvironmentsTearDownEnd(const UnitTest& /*unit_test*/) {}
01113     virtual void OnTestIterationEnd(const UnitTest& /*unit_test*/,
01114                                     int /*iteration*/) {}
01115     virtual void OnTestProgramEnd(const UnitTest& /*unit_test*/) {}
01116 };
01117
01118 // TestEventListeners lets users add listeners to track events in Google Test.
01119 class GTEST_API_ TestEventListeners {
01120 public:
01121     TestEventListeners();
01122     ~TestEventListeners();
01123
01124     // Appends an event listener to the end of the list. Google Test assumes
01125     // the ownership of the listener (i.e. it will delete the listener when
01126     // the test program finishes).
01127     void Append(TestEventListener* listener);
01128
01129     // Removes the given event listener from the list and returns it. It then
01130     // becomes the caller's responsibility to delete the listener. Returns
01131     // NULL if the listener is not found in the list.
01132     TestEventListener* Release(TestEventListener* listener);
01133
01134     // Returns the standard listener responsible for the default console
01135     // output. Can be removed from the listeners list to shut down default
01136     // console output. Note that removing this object from the listener list
01137     // with Release transfers its ownership to the caller and makes this
01138     // function return NULL the next time.
01139     TestEventListener* default_result_printer() const {
01140         return default_result_printer_;
01141     }
01142
01143     // Returns the standard listener responsible for the default XML output
01144     // controlled by the --gtest_output=xml flag. Can be removed from the
01145     // listeners list by users who want to shut down the default XML output
01146     // controlled by this flag and substitute it with custom one. Note that
01147     // removing this object from the listener list with Release transfers its
01148     // ownership to the caller and makes this function return NULL the next
01149     // time.
01150     TestEventListener* default_xml_generator() const {
01151         return default_xml_generator_;
01152     }
01153
01154 private:
01155     friend class TestCase;
01156     friend class TestInfo;
01157     friend class internal::DefaultGlobalTestPartResultReporter;
01158     friend class internal::NoExecDeathTest;
01159     friend class internal::TestEventListenersAccessor;
01160     friend class internal::UnitTestImpl;
01161
01162     // Returns repeater that broadcasts the TestEventListener events to all
01163     // subscribers.
01164     TestEventListener* repeater();
01165
01166     // Sets the default_result_printer attribute to the provided listener.
01167     // The listener is also added to the listener list and previous
01168     // default_result_printer is removed from it and deleted. The listener can
01169     // also be NULL in which case it will not be added to the list. Does
01170     // nothing if the previous and the current listener objects are the same.
01171     void SetDefaultResultPrinter(TestEventListener* listener);
01172
01173     // Sets the default_xml_generator attribute to the provided listener. The
01174     // listener is also added to the listener list and previous
01175     // default_xml_generator is removed from it and deleted. The listener can
01176     // also be NULL in which case it will not be added to the list. Does
01177     // nothing if the previous and the current listener objects are the same.
01178     void SetDefaultXmlGenerator(TestEventListener* listener);
01179
01180     // Controls whether events will be forwarded by the repeater to the
01181     // listeners in the list.
01182     bool EventForwardingEnabled() const;
01183     void SuppressEventForwarding();
01184
01185     // The actual list of listeners.
01186     internal::TestEventRepeater* repeater_;
01187     // Listener responsible for the standard result output.
01188     TestEventListener* default_result_printer_;
01189     // Listener responsible for the creation of the XML output file.

```

```
01190     TestEventListener* default_xml_generator_;
01191
01192     // We disallow copying TestEventListeners.
01193     GTEST_DISALLOW_COPY_AND_ASSIGN_(TestEventListeners);
01194 };
01195
01196 // A UnitTest consists of a vector of TestCases.
01197 //
01198 // This is a singleton class. The only instance of UnitTest is
01199 // created when UnitTest::GetInstance() is first called. This
01200 // instance is never deleted.
01201 //
01202 // UnitTest is not copyable.
01203 //
01204 // This class is thread-safe as long as the methods are called
01205 // according to their specification.
01206 class GTEST_API_ UnitTest {
01207 public:
01208     // Gets the singleton UnitTest object. The first time this method
01209     // is called, a UnitTest object is constructed and returned.
01210     // Consecutive calls will return the same object.
01211     static UnitTest* GetInstance();
01212
01213     // Runs all tests in this UnitTest object and prints the result.
01214     // Returns 0 if successful, or 1 otherwise.
01215     //
01216     // This method can only be called from the main thread.
01217     //
01218     // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
01219     int Run() GTEST_MUST_USE_RESULT_;
01220
01221     // Returns the working directory when the first TEST() or TEST_F()
01222     // was executed. The UnitTest object owns the string.
01223     const char* original_working_dir() const;
01224
01225     // Returns the TestCase object for the test that's currently running,
01226     // or NULL if no test is running.
01227     const TestCase* current_test_case() const
01228         GTEST_LOCK_EXCLUDED_(mutex_);
01229
01230     // Returns the TestInfo object for the test that's currently running,
01231     // or NULL if no test is running.
01232     const TestInfo* current_test_info() const
01233         GTEST_LOCK_EXCLUDED_(mutex_);
01234
01235     // Returns the random seed used at the start of the current test run.
01236     int random_seed() const;
01237
01238     // Returns the ParameterizedTestCaseRegistry object used to keep track of
01239     // value-parameterized tests and instantiate and register them.
01240     //
01241     // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
01242     internal::ParameterizedTestCaseRegistry& parameterized_test_registry()
01243         GTEST_LOCK_EXCLUDED_(mutex_);
01244
01245     // Gets the number of successful test cases.
01246     int successful_test_case_count() const;
01247
01248     // Gets the number of failed test cases.
01249     int failed_test_case_count() const;
01250
01251     // Gets the number of all test cases.
01252     int total_test_case_count() const;
01253
01254     // Gets the number of all test cases that contain at least one test
01255     // that should run.
01256     int test_case_to_run_count() const;
01257
01258     // Gets the number of successful tests.
01259     int successful_test_count() const;
01260
01261     // Gets the number of failed tests.
01262     int failed_test_count() const;
01263
01264     // Gets the number of disabled tests that will be reported in the XML report.
01265     int reportable_disabled_test_count() const;
01266
01267     // Gets the number of disabled tests.
01268     int disabled_test_count() const;
01269
01270     // Gets the number of tests to be printed in the XML report.
01271     int reportable_test_count() const;
01272
01273     // Gets the number of all tests.
01274     int total_test_count() const;
01275
01276     // Gets the number of tests that should run.
```

```

01277 int test_to_run_count() const;
01278 // Gets the time of the test program start, in ms from the start of the
01279 // UNIX epoch.
01280 TimeInMillis start_timestamp() const;
01282
01283 // Gets the elapsed time, in milliseconds.
01284 TimeInMillis elapsed_time() const;
01285
01286 // Returns true iff the unit test passed (i.e. all test cases passed).
01287 bool Passed() const;
01288
01289 // Returns true iff the unit test failed (i.e. some test case failed
01290 // or something outside of all tests failed).
01291 bool Failed() const;
01292
01293 // Gets the i-th test case among all the test cases. i can range from 0 to
01294 // total_test_case_count() - 1. If i is not in that range, returns NULL.
01295 const TestCase* GetTestCase(int i) const;
01296
01297 // Returns the TestResult containing information on test failures and
01298 // properties logged outside of individual test cases.
01299 const TestResult& ad_hoc_test_result() const;
01300
01301 // Returns the list of event listeners that can be used to track events
01302 // inside Google Test.
01303 TestEventListeners& listeners();
01304
01305 private:
01306 // Registers and returns a global test environment. When a test
01307 // program is run, all global test environments will be set-up in
01308 // the order they were registered. After all tests in the program
01309 // have finished, all global test environments will be torn-down in
01310 // the *reverse* order they were registered.
01311 //
01312 // The UnitTest object takes ownership of the given environment.
01313 //
01314 // This method can only be called from the main thread.
01315 Environment* AddEnvironment(Environment* env);
01316
01317 // Adds a TestPartResult to the current TestResult object. All
01318 // Google Test assertion macros (e.g. ASSERT_TRUE, EXPECT_EQ, etc)
01319 // eventually call this to report their results. The user code
01320 // should use the assertion macros instead of calling this directly.
01321 void AddTestPartResult(TestPartResult::Type result_type,
01322 const char* file_name,
01323 int line_number,
01324 const std::string& message,
01325 const std::string& os_stack_trace)
01326     GTEST_LOCK_EXCLUDED_(mutex_);
01327
01328 // Adds a TestProperty to the current TestResult object when invoked from
01329 // inside a test, to current TestCase's ad_hoc_test_result_ when invoked
01330 // from SetUpTestCase or TearDownTestCase, or to the global property set
01331 // when invoked elsewhere. If the result already contains a property with
01332 // the same key, the value will be updated.
01333 void RecordProperty(const std::string& key, const std::string& value);
01334
01335 // Gets the i-th test case among all the test cases. i can range from 0 to
01336 // total_test_case_count() - 1. If i is not in that range, returns NULL.
01337 TestCase* GetMutableTestCase(int i);
01338
01339 // Accessors for the implementation object.
01340 internal::UnitTestImpl* impl() { return impl_; }
01341 const internal::UnitTestImpl* impl() const { return impl_; }
01342
01343 // These classes and functions are friends as they need to access private
01344 // members of UnitTest.
01345 friend class ScopedTrace;
01346 friend class Test;
01347 friend class internal::AssertHelper;
01348 friend class internal::StreamingListenerTest;
01349 friend class internal::UnitTestRecordPropertyTestHelper;
01350 friend Environment* AddGlobalTestEnvironment(Environment* env);
01351 friend internal::UnitTestImpl* internal::GetUnitTestImpl();
01352 friend void internal::ReportFailureInUnknownLocation(
01353     TestPartResult::Type result_type,
01354     const std::string& message);
01355
01356 // Creates an empty UnitTest.
01357 UnitTest();
01358
01359 // D'tor
01360 virtual ~UnitTest();
01361
01362 // Pushes a trace defined by SCOPED_TRACE() on to the per-thread
01363 // Google Test trace stack.

```

```
01364     void PushGTestTrace(const internal::TraceInfo& trace)
01365         GTEST_LOCK_EXCLUDED_(mutex_);
01366
01367     // Pops a trace from the per-thread Google Test trace stack.
01368     void PopGTestTrace()
01369         GTEST_LOCK_EXCLUDED_(mutex_);
01370
01371     // Protects mutable state in *impl_. This is mutable as some const
01372     // methods need to lock it too.
01373     mutable internal::Mutex mutex_;
01374
01375     // Opaque implementation object. This field is never changed once
01376     // the object is constructed. We don't mark it as const here, as
01377     // doing so will cause a warning in the constructor of UnitTest.
01378     // Mutable state in *impl_ is protected by mutex_.
01379     internal::UnitTestImpl* impl_;
01380
01381     // We disallow copying UnitTest.
01382     GTEST_DISALLOW_COPY_AND_ASSIGN_(UnitTest);
01383 }
01384
01385 // A convenient wrapper for adding an environment for the test
01386 // program.
01387 //
01388 // You should call this before RUN_ALL_TESTS() is called, probably in
01389 // main(). If you use gtest_main, you need to call this before main()
01390 // starts for it to take effect. For example, you can define a global
01391 // variable like this:
01392 //
01393 //     testing::Environment* const foo_env =
01394 //         testing::AddGlobalTestEnvironment(new FooEnvironment);
01395 //
01396 // However, we strongly recommend you to write your own main() and
01397 // call AddGlobalTestEnvironment() there, as relying on initialization
01398 // of global variables makes the code harder to read and may cause
01399 // problems when you register multiple environments from different
01400 // translation units and the environments have dependencies among them
01401 // (remember that the compiler doesn't guarantee the order in which
01402 // global variables from different translation units are initialized).
01403 inline Environment* AddGlobalTestEnvironment(Environment* env) {
01404     return UnitTest::GetInstance()->AddEnvironment(env);
01405 }
01406
01407 // Initializes Google Test. This must be called before calling
01408 // RUN_ALL_TESTS(). In particular, it parses a command line for the
01409 // flags that Google Test recognizes. Whenever a Google Test flag is
01410 // seen, it is removed from argv, and *argc is decremented.
01411 //
01412 // No value is returned. Instead, the Google Test flag variables are
01413 // updated.
01414 //
01415 // Calling the function for the second time has no user-visible effect.
01416 GTEST_API_ void InitGoogleTest(int* argc, char** argv);
01417
01418 // This overloaded version can be used in Windows programs compiled in
01419 // UNICODE mode.
01420 GTEST_API_ void InitGoogleTest(int* argc, wchar_t** argv);
01421
01422 namespace internal {
01423
01424 // Separate the error generating code from the code path to reduce the stack
01425 // frame size of CmpHelperEQ. This helps reduce the overhead of some sanitizers
01426 // when calling EXPECT_* in a tight loop.
01427 template <typename T1, typename T2>
01428 AssertionResult CmpHelperEQFailure(const char* lhs_expression,
01429                                     const char* rhs_expression,
01430                                     const T1& lhs, const T2& rhs) {
01431     return EqFailure(lhs_expression,
01432                      rhs_expression,
01433                      FormatForComparisonFailureMessage(lhs, rhs),
01434                      FormatForComparisonFailureMessage(rhs, lhs),
01435                      false);
01436 }
01437
01438 // The helper function for {ASSERT|EXPECT}_EQ.
01439 template <typename T1, typename T2>
01440 AssertionResult CmpHelperEQ(const char* lhs_expression,
01441                             const char* rhs_expression,
01442                             const T1& lhs,
01443                             const T2& rhs) {
01444     if (lhs == rhs) {
01445         return AssertionSuccess();
01446     }
01447
01448     return CmpHelperEQFailure(lhs_expression, rhs_expression, lhs, rhs);
01449 }
```

```

01451 // With this overloaded version, we allow anonymous enums to be used
01452 // in {ASSERT|EXPECT}_EQ when compiled with gcc 4, as anonymous enums
01453 // can be implicitly cast to BiggestInt.
01454 GTEST_API_ AssertionResult CmpHelperEQ(const char* lhs_expression,
01455                                         const char* rhs_expression,
01456                                         BiggestInt lhs,
01457                                         BiggestInt rhs);
01458
01459 // The helper class for {ASSERT|EXPECT}_EQ. The template argument
01460 // lhs_is_null_literal is true iff the first argument to ASSERT_EQ()
01461 // is a null pointer literal. The following default implementation is
01462 // for lhs_is_null_literal being false.
01463 template <bool lhs_is_null_literal>
01464 class EqHelper {
01465 public:
01466   // This templatized version is for the general case.
01467   template <typename T1, typename T2>
01468   static AssertionResult Compare(const char* lhs_expression,
01469                                 const char* rhs_expression,
01470                                 const T1& lhs,
01471                                 const T2& rhs) {
01472     return CmpHelperEQ(lhs_expression, rhs_expression, lhs, rhs);
01473   }
01474
01475   // With this overloaded version, we allow anonymous enums to be used
01476   // in {ASSERT|EXPECT}_EQ when compiled with gcc 4, as anonymous
01477   // enums can be implicitly cast to BiggestInt.
01478   //
01479   // Even though its body looks the same as the above version, we
01480   // cannot merge the two, as it will make anonymous enums unhappy.
01481   static AssertionResult Compare(const char* lhs_expression,
01482                                 const char* rhs_expression,
01483                                 BiggestInt lhs,
01484                                 BiggestInt rhs) {
01485     return CmpHelperEQ(lhs_expression, rhs_expression, lhs, rhs);
01486   }
01487 };
01488
01489 // This specialization is used when the first argument to ASSERT_EQ()
01490 // is a null pointer literal, like NULL, false, or 0.
01491 template <>
01492 class EqHelper<true> {
01493 public:
01494   // We define two overloaded versions of Compare(). The first
01495   // version will be picked when the second argument to ASSERT_EQ() is
01496   // NOT a pointer, e.g. ASSERT_EQ(0, AnIntFunction()) or
01497   // EXPECT_EQ(false, a_bool).
01498   template <typename T1, typename T2>
01499   static AssertionResult Compare(
01500     const char* lhs_expression,
01501     const char* rhs_expression,
01502     const T1& lhs,
01503     const T2& rhs,
01504     // The following line prevents this overload from being considered if T2
01505     // is not a pointer type. We need this because ASSERT_EQ(NULL, my_ptr)
01506     // expands to Compare("", "", NULL, my_ptr), which requires a conversion
01507     // to match the Secret* in the other overload, which would otherwise make
01508     // this template match better.
01509     typename EnableIf<!is_pointer<T2>::value>::type* = 0) {
01510     return CmpHelperEQ(lhs_expression, rhs_expression, lhs, rhs);
01511   }
01512
01513   // This version will be picked when the second argument to ASSERT_EQ() is a
01514   // pointer, e.g. ASSERT_EQ(NULL, a_pointer).
01515   template <typename T>
01516   static AssertionResult Compare(
01517     const char* lhs_expression,
01518     const char* rhs_expression,
01519     // We used to have a second template parameter instead of Secret*. That
01520     // template parameter would deduce to 'long', making this a better match
01521     // than the first overload even without the first overload's EnableIf.
01522     // Unfortunately, gcc with -Wconversion-null warns when "passing NULL to
01523     // non-pointer argument" (even a deduced integral argument), so the old
01524     // implementation caused warnings in user code.
01525     Secret* /* lhs (NULL) */,
01526     T* rhs) {
01527     // We already know that 'lhs' is a null pointer.
01528     return CmpHelperEQ(lhs_expression, rhs_expression,
01529                         static_cast<T*>(NULL), rhs);
01530   }
01531 };
01532
01533 // Separate the error generating code from the code path to reduce the stack
01534 // frame size of CmpHelperOP. This helps reduce the overhead of some sanitizers
01535 // when calling EXPECT_OP in a tight loop.
01536 template <typename T1, typename T2>
01537 AssertionResult CmpHelperOpFailure(const char* expr1, const char* expr2,

```

```
01538             const T1& val1, const T2& val2,
01539             const char* op) {
01540     return AssertionFailure()
01541     << "Expected: (" << expr1 << ") " << op << " (" << expr2
01542     << "), actual: " << FormatForComparisonFailureMessage(val1, val2)
01543     << " vs " << FormatForComparisonFailureMessage(val2, val1);
01544 }
01545
01546 // A macro for implementing the helper functions needed to implement
01547 // ASSERT_?? and EXPECT_?. It is here just to avoid copy-and-paste
01548 // of similar code.
01549 //
01550 // For each templated helper function, we also define an overloaded
01551 // version for BiggestInt in order to reduce code bloat and allow
01552 // anonymous enums to be used with {ASSERT|EXPECT}_?? when compiled
01553 // with gcc 4.
01554 //
01555 // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
01556
01557 #define GTEST_IMPL_CMP_HELPER_(op_name, op) \
01558 template <typename T1, typename T2> \
01559 AssertionResult CmpHelper##op_name(const char* expr1, const char* expr2, \
01560                                     const T1& val1, const T2& val2) { \
01561     if (val1 op val2) { \
01562         return AssertionSuccess(); \
01563     } else { \
01564         return CmpHelperOpFailure(expr1, expr2, val1, val2, #op); \
01565     } \
01566 } \
01567 GTEST_API_ AssertionResult CmpHelper##op_name( \
01568     const char* expr1, const char* expr2, BiggestInt val1, BiggestInt val2) \
01569
01570 // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
01571
01572 // Implements the helper function for {ASSERT|EXPECT}_NE
01573 GTEST_IMPL_CMP_HELPER_(NE, !=);
01574 // Implements the helper function for {ASSERT|EXPECT}_LE
01575 GTEST_IMPL_CMP_HELPER_(LE, <=);
01576 // Implements the helper function for {ASSERT|EXPECT}_LT
01577 GTEST_IMPL_CMP_HELPER_(LT, <);
01578 // Implements the helper function for {ASSERT|EXPECT}_GE
01579 GTEST_IMPL_CMP_HELPER_(GE, >=);
01580 // Implements the helper function for {ASSERT|EXPECT}_GT
01581 GTEST_IMPL_CMP_HELPER_(GT, >);
01582
01583 #undef GTEST_IMPL_CMP_HELPER_
01584
01585 // The helper function for {ASSERT|EXPECT}_STREQ.
01586 //
01587 // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
01588 GTEST_API_ AssertionResult CmpHelperSTREQ(const char* s1_expression,
01589                                             const char* s2_expression,
01590                                             const char* s1,
01591                                             const char* s2);
01592
01593 // The helper function for {ASSERT|EXPECT}_STRCASEEQ.
01594 //
01595 // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
01596 GTEST_API_ AssertionResult CmpHelperSTRCASEEQ(const char* s1_expression,
01597                                                 const char* s2_expression,
01598                                                 const char* s1,
01599                                                 const char* s2);
01600
01601 // The helper function for {ASSERT|EXPECT}_STRNE.
01602 //
01603 // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
01604 GTEST_API_ AssertionResult CmpHelperSTRNE(const char* s1_expression,
01605                                             const char* s2_expression,
01606                                             const char* s1,
01607                                             const char* s2);
01608
01609 // The helper function for {ASSERT|EXPECT}_STRCASENE.
01610 //
01611 // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
01612 GTEST_API_ AssertionResult CmpHelperSTRCASENE(const char* s1_expression,
01613                                                 const char* s2_expression,
01614                                                 const char* s1,
01615                                                 const char* s2);
01616
01617
01618 // Helper function for *_STREQ on wide strings.
01619 //
01620 // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
01621 GTEST_API_ AssertionResult CmpHelperSTREQ(const char* s1_expression,
01622                                             const char* s2_expression,
01623                                             const wchar_t* s1,
01624                                             const wchar_t* s2);
```

```

01625 // Helper function for *_STRNE on wide strings.
01626 //
01627 // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
01628 GTEST_API_ AssertionResult CmpHelperSTRNE(const char* s1_expression,
01629                                                 const char* s2_expression,
01630                                                 const wchar_t* s1,
01631                                                 const wchar_t* s2);
01632
01633 } // namespace internal
01635
01636 // IsSubstring() and IsNotSubstring() are intended to be used as the
01637 // first argument to {EXPECT,ASSERT}_PRED_FORMAT2(), not by
01638 // themselves. They check whether needle is a substring of haystack
01639 // (NULL is considered a substring of itself only), and return an
01640 // appropriate error message when they fail.
01641 //
01642 // The {needle,haystack}_expr arguments are the stringified
01643 // expressions that generated the two real arguments.
01644 GTEST_API_ AssertionResult IsSubstring(
01645     const char* needle_expr, const char* haystack_expr,
01646     const char* needle, const char* haystack);
01647 GTEST_API_ AssertionResult IsNotSubstring(
01648     const char* needle_expr, const char* haystack_expr,
01649     const wchar_t* needle, const wchar_t* haystack);
01650 GTEST_API_ AssertionResult IsNotSubstring(
01651     const char* needle_expr, const char* haystack_expr,
01652     const char* needle, const char* haystack);
01653 GTEST_API_ AssertionResult IsNotSubstring(
01654     const char* needle_expr, const char* haystack_expr,
01655     const wchar_t* needle, const wchar_t* haystack);
01656 GTEST_API_ AssertionResult IsSubstring(
01657     const char* needle_expr, const char* haystack_expr,
01658     const ::std::string& needle, const ::std::string& haystack);
01659 GTEST_API_ AssertionResult IsNotSubstring(
01660     const char* needle_expr, const char* haystack_expr,
01661     const ::std::string& needle, const ::std::string& haystack);
01662
01663 #if GTEST_HAS_STD_WSTRING
01664 GTEST_API_ AssertionResult IsSubstring(
01665     const char* needle_expr, const char* haystack_expr,
01666     const ::std::wstring& needle, const ::std::wstring& haystack);
01667 GTEST_API_ AssertionResult IsNotSubstring(
01668     const char* needle_expr, const char* haystack_expr,
01669     const ::std::wstring& needle, const ::std::wstring& haystack);
01670 #endif // GTEST_HAS_STD_WSTRING
01671
01672 namespace internal {
01673
01674 // Helper template function for comparing floating-points.
01675 //
01676 // Template parameter:
01677 //
01678 // RawType: the raw floating-point type (either float or double)
01679 //
01680 // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
01681 template <typename RawType>
01682 AssertionResult CmpHelperFloatingPointEQ(const char* lhs_expression,
01683                                             const char* rhs_expression,
01684                                             RawType lhs_value,
01685                                             RawType rhs_value) {
01686     const FloatingPoint<RawType> lhs(lhs_value), rhs(rhs_value);
01687
01688     if (lhs.AlmostEquals(rhs)) {
01689         return AssertionSuccess();
01690     }
01691
01692     ::std::stringstream lhs_ss;
01693     lhs_ss << std::setprecision(std::numeric_limits<RawType>::digits10 + 2)
01694             << lhs_value;
01695
01696     ::std::stringstream rhs_ss;
01697     rhs_ss << std::setprecision(std::numeric_limits<RawType>::digits10 + 2)
01698             << rhs_value;
01699
01700     return EqFailure(lhs_expression,
01701                     rhs_expression,
01702                     StringStreamToString(&lhs_ss),
01703                     StringStreamToString(&rhs_ss),
01704                     false);
01705 }
01706
01707 // Helper function for implementing ASSERT_NEAR.
01708 //
01709 // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
01710 GTEST_API_ AssertionResult DoubleNearPredFormat(const char* expr1,
01711                                                 const char* expr2,

```

```
01712                                     const char* abs_error_expr,
01713                                     double val1,
01714                                     double val2,
01715                                     double abs_error);
01716
01717 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
01718 // A class that enables one to stream messages to assertion macros
01719 class GTEST_API_ AssertHelper {
01720 public:
01721     // Constructor.
01722     AssertHelper(TestPartResult::Type type,
01723                  const char* file,
01724                  int line,
01725                  const char* message);
01726     ~AssertHelper();
01727
01728     // Message assignment is a semantic trick to enable assertion
01729     // streaming; see the GTEST_MESSAGE_ macro below.
01730     void operator=(const Message& message) const;
01731
01732 private:
01733     // We put our data in a struct so that the size of the AssertHelper class can
01734     // be as small as possible. This is important because gcc is incapable of
01735     // re-using stack space even for temporary variables, so every EXPECT_EQ
01736     // reserves stack space for another AssertHelper.
01737     struct AssertHelperData {
01738         AssertHelperData(TestPartResult::Type t,
01739                         const char* srcfile,
01740                         int line_num,
01741                         const char* msg)
01742             : type(t), file(srcfile), line(line_num), message(msg) { }
01743
01744     TestPartResult::Type const type;
01745     const char* const file;
01746     int const line;
01747     std::string const message;
01748
01749 private:
01750     GTEST_DISALLOW_COPY_AND_ASSIGN_(AssertHelperData);
01751 };
01752
01753     AssertHelperData* const data_;
01754
01755     GTEST_DISALLOW_COPY_AND_ASSIGN_(AssertHelper);
01756 };
01757
01758 } // namespace internal
01759
01760 // The pure interface class that all value-parameterized tests inherit from.
01761 // A value-parameterized class must inherit from both ::testing::Test and
01762 // ::testing::WithParamInterface. In most cases that just means inheriting
01763 // from ::testing::TestWithParam, but more complicated test hierarchies
01764 // may need to inherit from Test and WithParamInterface at different levels.
01765 //
01766 // This interface has support for accessing the test parameter value via
01767 // the GetParam() method.
01768 //
01769 // Use it with one of the parameter generator defining functions, like Range(),
01770 // Values(), ValuesIn(), Bool(), and Combine().
01771 //
01772 // class FooTest : public ::testing::TestWithParam<int> {
01773 // protected:
01774 //     FooTest() {
01775 //         // Can use GetParam() here.
01776 //     }
01777 //     virtual ~FooTest() {
01778 //         // Can use GetParam() here.
01779 //     }
01780 //     virtual void SetUp() {
01781 //         // Can use GetParam() here.
01782 //     }
01783 //     virtual void TearDown() {
01784 //         // Can use GetParam() here.
01785 //     }
01786 // };
01787 // TEST_P(FooTest, DoesBar) {
01788 //     // Can use GetParam() method here.
01789 //     Foo foo;
01790 //     ASSERT_TRUE(foo.DoesBar(GetParam()));
01791 // }
01792 // INSTANTIATE_TEST_CASE_P(OneToTenRange, FooTest, ::testing::Range(1, 10));
01793
01794 template <typename T>
01795 class WithParamInterface {
01796 public:
01797     typedef T ParamType;
01798     virtual ~WithParamInterface() {}
```

```

01799 // The current parameter value. Is also available in the test fixture's
01800 // constructor. This member function is non-static, even though it only
01801 // references static data, to reduce the opportunity for incorrect uses
01802 // like writing 'WithParamInterface<bool>::GetParam()' for a test that
01803 // uses a fixture whose parameter type is int.
01804 const ParamType& GetParam() const {
01805     GTEST_CHECK_(parameter_ != NULL)
01806     << "GetParam() can only be called inside a value-parameterized test "
01807     << "-- did you intend to write TEST_P instead of TEST_F?";
01808     return *parameter_;
01809 }
01810 }
01811
01812 private:
01813 // Sets parameter value. The caller is responsible for making sure the value
01814 // remains alive and unchanged throughout the current test.
01815 static void SetParam(const ParamType* parameter) {
01816     parameter_ = parameter;
01817 }
01818
01819 // Static value used for accessing parameter during a test lifetime.
01820 static const ParamType* parameter_;
01821
01822 // TestClass must be a subclass of WithParamInterface<T> and Test.
01823 template <class TestClass> friend class internal::ParameterizedTestFactory;
01824 };
01825
01826 template <typename T>
01827 const T* WithParamInterface<T>::parameter_ = NULL;
01828
01829 // Most value-parameterized classes can ignore the existence of
01830 // WithParamInterface, and can just inherit from ::testing::TestWithParam.
01831
01832 template <typename T>
01833 class TestWithParam : public Test, public WithParamInterface<T> {
01834 };
01835
01836 // Macros for indicating success/failure in test code.
01837
01838 // ADD_FAILURE unconditionally adds a failure to the current test.
01839 // SUCCEED generates a success - it doesn't automatically make the
01840 // current test successful, as a test is only successful when it has
01841 // no failure.
01842 //
01843 // EXPECT_* verifies that a certain condition is satisfied. If not,
01844 // it behaves like ADD_FAILURE. In particular:
01845 //
01846 //   EXPECT_TRUE verifies that a Boolean condition is true.
01847 //   EXPECT_FALSE verifies that a Boolean condition is false.
01848 //
01849 // FAIL and ASSERT_* are similar to ADD_FAILURE and EXPECT_*, except
01850 // that they will also abort the current function on failure. People
01851 // usually want the fail-fast behavior of FAIL and ASSERT_*, but those
01852 // writing data-driven tests often find themselves using ADD_FAILURE
01853 // and EXPECT_* more.
01854
01855 // Generates a nonfatal failure with a generic message.
01856 #define ADD_FAILURE() GTEST_NONFATAL_FAILURE_("Failed")
01857
01858 // Generates a nonfatal failure at the given source file location with
01859 // a generic message.
01860 #define ADD_FAILURE_AT(file, line) \
01861     GTEST_MESSAGE_AT_(file, line, "Failed", \
01862                         ::testing::TestPartResult::kNonFatalFailure)
01863
01864 // Generates a fatal failure with a generic message.
01865 #define GTEST_FAIL() GTEST_FATAL_FAILURE_("Failed")
01866
01867 // Define this macro to 1 to omit the definition of FAIL(), which is a
01868 // generic name and clashes with some other libraries.
01869 #if !GTEST_DONT_DEFINE_FAIL
01870 # define FAIL() GTEST_FAIL()
01871 #endif
01872
01873 // Generates a success with a generic message.
01874 #define GTEST_SUCCEED() GTEST_SUCCESS_("Succeeded")
01875
01876 // Define this macro to 1 to omit the definition of SUCCEED(), which
01877 // is a generic name and clashes with some other libraries.
01878 #if !GTEST_DONT_DEFINE_SUCCEED
01879 # define SUCCEED() GTEST_SUCCEED()
01880 #endif
01881
01882 // Macros for testing exceptions.
01883 //
01884 //   * {ASSERT|EXPECT}_THROW(statement, expected_exception):
01885 //           Tests that the statement throws the expected exception.

```

```

01886 //      * {ASSERT|EXPECT}_NO_THROW(statement):
01887 //          Tests that the statement doesn't throw any exception.
01888 //      * {ASSERT|EXPECT}_ANY_THROW(statement):
01889 //          Tests that the statement throws an exception.
01890
01891 #define EXPECT_THROW(statement, expected_exception) \
01892     GTEST_TEST_THROW_(statement, expected_exception, GTEST_NONFATAL_FAILURE_)
01893 #define EXPECT_NO_THROW(statement) \
01894     GTEST_TEST_NO_THROW_(statement, GTEST_NONFATAL_FAILURE_)
01895 #define EXPECT_ANY_THROW(statement) \
01896     GTEST_TEST_ANY_THROW_(statement, GTEST_NONFATAL_FAILURE_)
01897 #define ASSERT_THROW(statement, expected_exception) \
01898     GTEST_TEST_THROW_(statement, expected_exception, GTEST_FATAL_FAILURE_)
01899 #define ASSERT_NO_THROW(statement) \
01900     GTEST_TEST_NO_THROW_(statement, GTEST_FATAL_FAILURE_)
01901 #define ASSERT_ANY_THROW(statement) \
01902     GTEST_TEST_ANY_THROW_(statement, GTEST_FATAL_FAILURE_)
01903
01904 // Boolean assertions. Condition can be either a Boolean expression or an
01905 // AssertionResult. For more information on how to use AssertionResult with
01906 // these macros see comments on that class.
01907 #define EXPECT_TRUE(condition) \
01908     GTEST_TEST_BOOLEAN_(condition, #condition, false, true, \
01909         GTEST_NONFATAL_FAILURE_)
01910 #define EXPECT_FALSE(condition) \
01911     GTEST_TEST_BOOLEAN_(!(condition), #condition, true, false, \
01912         GTEST_NONFATAL_FAILURE_)
01913 #define ASSERT_TRUE(condition) \
01914     GTEST_TEST_BOOLEAN_(condition, #condition, false, true, \
01915         GTEST_FATAL_FAILURE_)
01916 #define ASSERT_FALSE(condition) \
01917     GTEST_TEST_BOOLEAN_(!(condition), #condition, true, false, \
01918         GTEST_FATAL_FAILURE_)
01919
01920 // Macros for testing equalities and inequalities.
01921 //
01922 //      * {ASSERT|EXPECT}_EQ(v1, v2): Tests that v1 == v2
01923 //      * {ASSERT|EXPECT}_NE(v1, v2): Tests that v1 != v2
01924 //      * {ASSERT|EXPECT}_LT(v1, v2): Tests that v1 < v2
01925 //      * {ASSERT|EXPECT}_LE(v1, v2): Tests that v1 <= v2
01926 //      * {ASSERT|EXPECT}_GT(v1, v2): Tests that v1 > v2
01927 //      * {ASSERT|EXPECT}_GE(v1, v2): Tests that v1 >= v2
01928 //
01929 // When they are not, Google Test prints both the tested expressions and
01930 // their actual values. The values must be compatible built-in types,
01931 // or you will get a compiler error. By "compatible" we mean that the
01932 // values can be compared by the respective operator.
01933 //
01934 // Note:
01935 //
01936 // 1. It is possible to make a user-defined type work with
01937 //     {ASSERT|EXPECT}_??(), but that requires overloading the
01938 //     comparison operators and is thus discouraged by the Google C++
01939 //     Usage Guide. Therefore, you are advised to use the
01940 //     {ASSERT|EXPECT}_TRUE() macro to assert that two objects are
01941 //     equal.
01942 //
01943 // 2. The {ASSERT|EXPECT}_??() macros do pointer comparisons on
01944 //     pointers (in particular, C strings). Therefore, if you use it
01945 //     with two C strings, you are testing how their locations in memory
01946 //     are related, not how their content is related. To compare two C
01947 //     strings by content, use {ASSERT|EXPECT}_STR*().
01948 //
01949 // 3. {ASSERT|EXPECT}_EQ(v1, v2) is preferred to
01950 //     {ASSERT|EXPECT}_TRUE(v1 == v2), as the former tells you
01951 //     what the actual value is when it fails, and similarly for the
01952 //     other comparisons.
01953 //
01954 // 4. Do not depend on the order in which {ASSERT|EXPECT}_??()
01955 //     evaluate their arguments, which is undefined.
01956 //
01957 // 5. These macros evaluate their arguments exactly once.
01958 //
01959 // Examples:
01960 //
01961 //     EXPECT_NE(Foo(), 5);
01962 //     EXPECT_EQ(a_pointer, NULL);
01963 //     ASSERT_LT(i, array_size);
01964 //     ASSERT_GT(records.size(), 0) << "There is no record left.";
01965
01966 #define EXPECT_EQ(val1, val2) \
01967     EXPECT_PRED_FORMAT2(::testing::internal:: \
01968         EqHelper<GTEST_IS_NULL_LITERAL_(val1)>::Compare, \
01969         val1, val2)
01970 #define EXPECT_NE(val1, val2) \
01971     EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperNE, val1, val2)
01972 #define EXPECT_LE(val1, val2) \

```

```

01973 EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperLE, val1, val2)
01974 #define EXPECT_LT(val1, val2) \
01975   EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperLT, val1, val2)
01976 #define EXPECT_GE(val1, val2) \
01977   EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperGE, val1, val2)
01978 #define EXPECT_GT(val1, val2) \
01979   EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperGT, val1, val2)
01980
01981 #define GTEST_ASSERT_EQ(val1, val2) \
01982   ASSERT_PRED_FORMAT2(::testing::internal:: \
01983     EqHelper<GTEST_IS_NULL_LITERAL_(val1)>::Compare, \
01984     val1, val2)
01985 #define GTEST_ASSERT_NE(val1, val2) \
01986   ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperNE, val1, val2)
01987 #define GTEST_ASSERT_LE(val1, val2) \
01988   ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperLE, val1, val2)
01989 #define GTEST_ASSERT_LT(val1, val2) \
01990   ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperLT, val1, val2)
01991 #define GTEST_ASSERT_GE(val1, val2) \
01992   ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperGE, val1, val2)
01993 #define GTEST_ASSERT_GT(val1, val2) \
01994   ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperGT, val1, val2)
01995
01996 // Define macro GTEST_DONT_DEFINE_ASSERT_XY to 1 to omit the definition of
01997 // ASSERT_XY(), which clashes with some users' own code.
01998
01999 #if !GTEST_DONT_DEFINE_ASSERT_EQ
02000 # define ASSERT_EQ(val1, val2) GTEST_ASSERT_EQ(val1, val2)
02001 #endif
02002
02003 #if !GTEST_DONT_DEFINE_ASSERT_NE
02004 # define ASSERT_NE(val1, val2) GTEST_ASSERT_NE(val1, val2)
02005 #endif
02006
02007 #if !GTEST_DONT_DEFINE_ASSERT_LE
02008 # define ASSERT_LE(val1, val2) GTEST_ASSERT_LE(val1, val2)
02009 #endif
02010
02011 #if !GTEST_DONT_DEFINE_ASSERT_LT
02012 # define ASSERT_LT(val1, val2) GTEST_ASSERT_LT(val1, val2)
02013 #endif
02014
02015 #if !GTEST_DONT_DEFINE_ASSERT_GE
02016 # define ASSERT_GE(val1, val2) GTEST_ASSERT_GE(val1, val2)
02017 #endif
02018
02019 #if !GTEST_DONT_DEFINE_ASSERT_GT
02020 # define ASSERT_GT(val1, val2) GTEST_ASSERT_GT(val1, val2)
02021 #endif
02022
02023 // C-string Comparisons. All tests treat NULL and any non-NUL string
02024 // as different. Two NULLs are equal.
02025 //
02026 // * {ASSERT|EXPECT}_STREQ(s1, s2): Tests that s1 == s2
02027 // * {ASSERT|EXPECT}_STRNE(s1, s2): Tests that s1 != s2
02028 // * {ASSERT|EXPECT}_STRCASEEQ(s1, s2): Tests that s1 == s2, ignoring case
02029 // * {ASSERT|EXPECT}_STRCASENE(s1, s2): Tests that s1 != s2, ignoring case
02030 //
02031 // For wide or narrow string objects, you can use the
02032 // {ASSERT|EXPECT}_??() macros.
02033 //
02034 // Don't depend on the order in which the arguments are evaluated,
02035 // which is undefined.
02036 //
02037 // These macros evaluate their arguments exactly once.
02038
02039 #define EXPECT_STREQ(s1, s2) \
02040   EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperSTREQ, s1, s2)
02041 #define EXPECT_STRNE(s1, s2) \
02042   EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperSTRNE, s1, s2)
02043 #define EXPECT_STRCASEEQ(s1, s2) \
02044   EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperSTRCASEEQ, s1, s2)
02045 #define EXPECT_STRCASENE(s1, s2) \
02046   EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperSTRCASENE, s1, s2)
02047
02048 #define ASSERT_STREQ(s1, s2) \
02049   ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperSTREQ, s1, s2)
02050 #define ASSERT_STRNE(s1, s2) \
02051   ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperSTRNE, s1, s2)
02052 #define ASSERT_STRCASEEQ(s1, s2) \
02053   ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperSTRCASEEQ, s1, s2)
02054 #define ASSERT_STRCASENE(s1, s2) \
02055   ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperSTRCASENE, s1, s2)
02056
02057 // Macros for comparing floating-point numbers.
02058 //
02059 // * {ASSERT|EXPECT}_FLOAT_EQ(val1, val2):

```

```

02060 //      Tests that two float values are almost equal.
02061 //      * {ASSERT|EXPECT}_DOUBLE_EQ(val1, val2):
02062 //      Tests that two double values are almost equal.
02063 //      * {ASSERT|EXPECT}_NEAR(v1, v2, abs_error):
02064 //      Tests that v1 and v2 are within the given distance to each other.
02065 //
02066 // Google Test uses ULP-based comparison to automatically pick a default
02067 // error bound that is appropriate for the operands. See the
02068 // FloatingPoint template class in gtest-internal.h if you are
02069 // interested in the implementation details.
02070
02071 #define EXPECT_FLOAT_EQ(val1, val2) \
02072     EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperFloatingPointEQ<float>, \
02073                         val1, val2)
02074
02075 #define EXPECT_DOUBLE_EQ(val1, val2) \
02076     EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperFloatingPointEQ<double>, \
02077                         val1, val2)
02078
02079 #define ASSERT_FLOAT_EQ(val1, val2) \
02080     ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperFloatingPointEQ<float>, \
02081                         val1, val2)
02082
02083 #define ASSERT_DOUBLE_EQ(val1, val2) \
02084     ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperFloatingPointEQ<double>, \
02085                         val1, val2)
02086
02087 #define EXPECT_NEAR(val1, val2, abs_error) \
02088     EXPECT_PRED_FORMAT3(::testing::internal::DoubleNearPredFormat, \
02089                         val1, val2, abs_error)
02090
02091 #define ASSERT_NEAR(val1, val2, abs_error) \
02092     ASSERT_PRED_FORMAT3(::testing::internal::DoubleNearPredFormat, \
02093                         val1, val2, abs_error)
02094
02095 // These predicate format functions work on floating-point values, and
02096 // can be used in {ASSERT|EXPECT}_PRED_FORMAT2*, e.g.
02097 //
02098 //     EXPECT_PRED_FORMAT2(testing::DoubleLE, Foo(), 5.0);
02099
02100 // Asserts that val1 is less than, or almost equal to, val2. Fails
02101 // otherwise. In particular, it fails if either val1 or val2 is NaN.
02102 GTEST_API_ AssertionResult FloatLE(const char* expr1, const char* expr2,
02103                                     float val1, float val2);
02104 GTEST_API_ AssertionResult DoubleLE(const char* expr1, const char* expr2,
02105                                     double val1, double val2);
02106
02107
02108 #if GTEST_OS_WINDOWS
02109
02110 // Macros that test for HRESULT failure and success, these are only useful
02111 // on Windows, and rely on Windows SDK macros and APIs to compile.
02112 //
02113 //      * {ASSERT|EXPECT}_HRESULT_{SUCCEEDED|FAILED}(expr)
02114 //
02115 // When expr unexpectedly fails or succeeds, Google Test prints the
02116 // expected result and the actual result with both a human-readable
02117 // string representation of the error, if available, as well as the
02118 // hex result code.
02119 # define EXPECT_HRESULT_SUCCEEDED(expr) \
02120     EXPECT_PRED_FORMAT1(::testing::internal::IsHRESULTSuccess, (expr))
02121
02122 # define ASSERT_HRESULT_SUCCEEDED(expr) \
02123     ASSERT_PRED_FORMAT1(::testing::internal::IsHRESULTSuccess, (expr))
02124
02125 # define EXPECT_HRESULT_FAILED(expr) \
02126     EXPECT_PRED_FORMAT1(::testing::internal::IsHRESULTFailure, (expr))
02127
02128 # define ASSERT_HRESULT_FAILED(expr) \
02129     ASSERT_PRED_FORMAT1(::testing::internal::IsHRESULTFailure, (expr))
02130
02131 #endif // GTEST_OS_WINDOWS
02132
02133 // Macros that execute statement and check that it doesn't generate new fatal
02134 // failures in the current thread.
02135 //
02136 //      * {ASSERT|EXPECT}_NO_FATAL_FAILURE(statement);
02137 //
02138 // Examples:
02139 //
02140 //      EXPECT_NO_FATAL_FAILURE(Process());
02141 //      ASSERT_NO_FATAL_FAILURE(Process()) << "Process() failed";
02142 //
02143 #define ASSERT_NO_FATAL_FAILURE(statement) \
02144     GTEST_TEST_NO_FATAL_FAILURE_(statement, GTEST_FATAL_FAILURE_)
02145 #define EXPECT_NO_FATAL_FAILURE(statement) \
02146     GTEST_TEST_NO_FATAL_FAILURE_(statement, GTEST_NONFATAL_FAILURE_)
```

```

02147
02148 // Causes a trace (including the given source file path and line number,
02149 // and the given message) to be included in every test failure message generated
02150 // by code in the scope of the lifetime of an instance of this class. The effect
02151 // is undone with the destruction of the instance.
02152 //
02153 // The message argument can be anything streamable to std::ostream.
02154 //
02155 // Example:
02156 //   testing::ScopedTrace trace("file.cc", 123, "message");
02157 //
02158 class GTEST_API_ ScopedTrace {
02159 public:
02160   // The c'tor pushes the given source file location and message onto
02161   // a trace stack maintained by Google Test.
02162
02163   // Template version. Uses Message() to convert the values into strings.
02164   // Slow, but flexible.
02165   template <typename T>
02166   ScopedTrace(const char* file, int line, const T& message) {
02167     PushTrace(file, line, (Message() << message).GetString());
02168   }
02169
02170   // Optimize for some known types.
02171   ScopedTrace(const char* file, int line, const char* message) {
02172     PushTrace(file, line, message ? message : "(null)");
02173   }
02174
02175 #if GTEST_HAS_GLOBAL_STRING
02176   ScopedTrace(const char* file, int line, const ::string& message) {
02177     PushTrace(file, line, message);
02178   }
02179 #endif
02180
02181   ScopedTrace(const char* file, int line, const std::string& message) {
02182     PushTrace(file, line, message);
02183   }
02184
02185   // The d'tor pops the info pushed by the c'tor.
02186   //
02187   // Note that the d'tor is not virtual in order to be efficient.
02188   // Don't inherit from ScopedTrace!
02189   ~ScopedTrace();
02190
02191 private:
02192   void PushTrace(const char* file, int line, std::string message);
02193
02194   GTEST_DISALLOW_COPY_AND_ASSIGN_(ScopedTrace);
02195 } GTEST_ATTRIBUTE_UNUSED_; // A ScopedTrace object does its job in its
02196 // c'tor and d'tor. Therefore it doesn't
02197 // need to be used otherwise.
02198
02199 // Causes a trace (including the source file path, the current line
02200 // number, and the given message) to be included in every test failure
02201 // message generated by code in the current scope. The effect is
02202 // undone when the control leaves the current scope.
02203 //
02204 // The message argument can be anything streamable to std::ostream.
02205 //
02206 // In the implementation, we include the current line number as part
02207 // of the dummy variable name, thus allowing multiple SCOPED_TRACE()'s
02208 // to appear in the same block - as long as they are on different
02209 // lines.
02210 //
02211 // Assuming that each thread maintains its own stack of traces.
02212 // Therefore, a SCOPED_TRACE() would (correctly) only affect the
02213 // assertions in its own thread.
02214 #define SCOPED_TRACE(message) \
02215   ::testing::ScopedTrace GTEST_CONCAT_TOKEN_(gtest_trace_, __LINE__)(\
02216   __FILE__, __LINE__, (message))
02217
02218
02219 // Compile-time assertion for type equality.
02220 // StaticAssertTypeEq<type1, type2>() compiles iff type1 and type2 are
02221 // the same type. The value it returns is not interesting.
02222 //
02223 // Instead of making StaticAssertTypeEq a class template, we make it a
02224 // function template that invokes a helper class template. This
02225 // prevents a user from misusing StaticAssertTypeEq<T1, T2> by
02226 // defining objects of that type.
02227 //
02228 // CAVEAT:
02229 //
02230 // When used inside a method of a class template,
02231 // StaticAssertTypeEq<T1, T2>() is effective ONLY IF the method is
02232 // instantiated. For example, given:
02233 //

```

```
02234 //     template <typename T> class Foo {
02235 //         public:
02236 //             void Bar() { testing::StaticAssertTypeEq<int, T>(); }
02237 //     };
02238 //
02239 // the code:
02240 //
02241 //     void Test1() { Foo<bool> foo; }
02242 //
02243 // will NOT generate a compiler error, as Foo<bool>::Bar() is never
02244 // actually instantiated. Instead, you need:
02245 //
02246 //     void Test2() { Foo<bool> foo; foo.Bar(); }
02247 //
02248 // to cause a compiler error.
02249 template <typename T1, typename T2>
02250 bool StaticAssertTypeEq() {
02251     (void)internal::StaticAssertTypeEqHelper<T1, T2>();
02252     return true;
02253 }
02254
02255 // Defines a test.
02256 //
02257 // The first parameter is the name of the test case, and the second
02258 // parameter is the name of the test within the test case.
02259 //
02260 // The convention is to end the test case name with "Test". For
02261 // example, a test case for the Foo class can be named FooTest.
02262 //
02263 // Test code should appear between braces after an invocation of
02264 // this macro. Example:
02265 //
02266 //     TEST(FooTest, InitializesCorrectly) {
02267 //         Foo foo;
02268 //         EXPECT_TRUE(foo.StatusIsOK());
02269 //     }
02270
02271 // Note that we call GetTestTypeId() instead of GetTypeId<
02272 // ::testing::Test>() here to get the type ID of testing::Test. This
02273 // is to work around a suspected linker bug when using Google Test as
02274 // a framework on Mac OS X. The bug causes GetTypeId<
02275 // ::testing::Test>() to return different values depending on whether
02276 // the call is from the Google Test framework itself or from user test
02277 // code. GetTestTypeId() is guaranteed to always return the same
02278 // value, as it always calls GetTypeId<>() from the Google Test
02279 // framework.
02280 #define GTEST_TEST(test_case_name, test_name) \
02281     GTEST_TEST_(test_case_name, test_name, \
02282                 ::testing::Test, ::testing::internal::GetTestTypeId())
02283
02284 // Define this macro to 1 to omit the definition of TEST(), which
02285 // is a generic name and clashes with some other libraries.
02286 #if !GTEST_DONT_DEFINE_TEST
02287 # define TEST(test_case_name, test_name) GTEST_TEST(test_case_name, test_name)
02288 #endif
02289
02290 // Defines a test that uses a test fixture.
02291 //
02292 // The first parameter is the name of the test fixture class, which
02293 // also doubles as the test case name. The second parameter is the
02294 // name of the test within the test case.
02295 //
02296 // A test fixture class must be declared earlier. The user should put
02297 // the test code between braces after using this macro. Example:
02298 //
02299 //     class FooTest : public testing::Test {
02300 //         protected:
02301 //             virtual void SetUp() { b_.AddElement(3); }
02302 //
02303 //             Foo a_;
02304 //             Foo b_;
02305 //         };
02306 //
02307 //     TEST_F(FooTest, InitializesCorrectly) {
02308 //         EXPECT_TRUE(a_.StatusIsOK());
02309 //     }
02310 //
02311 //     TEST_F(FooTest, ReturnsElementCountCorrectly) {
02312 //         EXPECT_EQ(a_.size(), 0);
02313 //         EXPECT_EQ(b_.size(), 1);
02314 //     }
02315
02316 #define TEST_F(fixture, test_name) \
02317     GTEST_TEST_(fixture, test_name, fixture, \
02318                 ::testing::internal::GetTypeId<fixture>())
02319
02320 // Returns a path to temporary directory.
```

```

02321 // Tries to determine an appropriate directory for the platform.
02322 GTEST_API std::string TempDir();
02323
02324 #ifdef _MSC_VER
02325 # pragma warning(pop)
02326 #endif
02327
02328 } // namespace testing
02329
02330 // Use this function in main() to run all tests. It returns 0 if all
02331 // tests are successful, or 1 otherwise.
02332 //
02333 // RUN_ALL_TESTS() should be invoked after the command line has been
02334 // parsed by InitGoogleTest().
02335 //
02336 // This function was formerly a macro; thus, it is in the global
02337 // namespace and has an all-caps name.
02338 int RUN_ALL_TESTS() GTEST_MUST_USE_RESULT_;
02339
02340 inline int RUN_ALL_TESTS() {
02341     return ::testing::UnitTest::GetInstance()->Run();
02342 }
02343
02344 GTEST_DISABLE_MSC_WARNINGS_POP_() // 4251
02345
02346 #endif // GTEST_INCLUDE_GTEST_GTEST_H_

```

9.22 Dokumentacja pliku

packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/custom/gtest.h

9.23 gtest.h

[Idź do dokumentacji tego pliku.](#)

```

00001 // Copyright 2015, Google Inc.
00002 // All rights reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //      * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //      * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 // Injection point for custom user configurations. See README for details
00031 //
00032 // ** Custom implementation starts here **
00033
00034 #ifndef GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_H_
00035 #define GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_H_
00036
00037 #endif // GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_H_

```

9.24 Dokumentacja pliku

packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest_pred_impl.h

```
#include "gtest/gtest.h"
```

Przestrzenie nazw

- namespace `testing`

Definicje

- `#define GTEST_ASSERT_(expression, on_failure)`
- `#define GTEST_PRED_FORMAT1_(pred_format, v1, on_failure)`
- `#define GTEST_PRED1_(pred, v1, on_failure)`
- `#define EXPECT_PRED_FORMAT1(pred_format, v1)`
- `#define EXPECT_PRED1(pred, v1)`
- `#define ASSERT_PRED_FORMAT1(pred_format, v1)`
- `#define ASSERT_PRED1(pred, v1)`
- `#define GTEST_PRED_FORMAT2_(pred_format, v1, v2, on_failure)`
- `#define GTEST_PRED2_(pred, v1, v2, on_failure)`
- `#define EXPECT_PRED_FORMAT2(pred_format, v1, v2)`
- `#define EXPECT_PRED2(pred, v1, v2)`
- `#define ASSERT_PRED_FORMAT2(pred_format, v1, v2)`
- `#define ASSERT_PRED2(pred, v1, v2)`
- `#define GTEST_PRED_FORMAT3_(pred_format, v1, v2, v3, on_failure)`
- `#define GTEST_PRED3_(pred, v1, v2, v3, on_failure)`
- `#define EXPECT_PRED_FORMAT3(pred_format, v1, v2, v3)`
- `#define EXPECT_PRED3(pred, v1, v2, v3)`
- `#define ASSERT_PRED_FORMAT3(pred_format, v1, v2, v3)`
- `#define ASSERT_PRED3(pred, v1, v2, v3)`
- `#define GTEST_PRED_FORMAT4_(pred_format, v1, v2, v3, v4, on_failure)`
- `#define GTEST_PRED4_(pred, v1, v2, v3, v4, on_failure)`
- `#define EXPECT_PRED_FORMAT4(pred_format, v1, v2, v3, v4)`
- `#define EXPECT_PRED4(pred, v1, v2, v3, v4)`
- `#define ASSERT_PRED_FORMAT4(pred_format, v1, v2, v3, v4)`
- `#define ASSERT_PRED4(pred, v1, v2, v3, v4)`
- `#define GTEST_PRED_FORMAT5_(pred_format, v1, v2, v3, v4, v5, on_failure)`
- `#define GTEST_PRED5_(pred, v1, v2, v3, v4, v5, on_failure)`
- `#define EXPECT_PRED_FORMAT5(pred_format, v1, v2, v3, v4, v5)`
- `#define EXPECT_PRED5(pred, v1, v2, v3, v4, v5)`
- `#define ASSERT_PRED_FORMAT5(pred_format, v1, v2, v3, v4, v5)`
- `#define ASSERT_PRED5(pred, v1, v2, v3, v4, v5)`

Funkcje

- template<typename Pred, typename T1>
AssertionResult **testing::AssertPred1Helper** (const char *pred_text, const char *e1, Pred pred, const T1 &v1)
- template<typename Pred, typename T1, typename T2>
AssertionResult **testing::AssertPred2Helper** (const char *pred_text, const char *e1, const char *e2, Pred pred, const T1 &v1, const T2 &v2)
- template<typename Pred, typename T1, typename T2, typename T3>
AssertionResult **testing::AssertPred3Helper** (const char *pred_text, const char *e1, const char *e2, const char *e3, Pred pred, const T1 &v1, const T2 &v2, const T3 &v3)
- template<typename Pred, typename T1, typename T2, typename T3, typename T4>
AssertionResult **testing::AssertPred4Helper** (const char *pred_text, const char *e1, const char *e2, const char *e3, const char *e4, Pred pred, const T1 &v1, const T2 &v2, const T3 &v3, const T4 &v4)
- template<typename Pred, typename T1, typename T2, typename T3, typename T4, typename T5>
AssertionResult **testing::AssertPred5Helper** (const char *pred_text, const char *e1, const char *e2, const char *e3, const char *e4, const char *e5, Pred pred, const T1 &v1, const T2 &v2, const T3 &v3, const T4 &v4, const T5 &v5)

9.24.1 Dokumentacja definicji

9.24.1.1 ASSERT_PRED1

```
#define ASSERT_PRED1(
    pred,
    v1)
```

Wartość:

```
GTEST_PRED1_(pred, v1, GTEST_FATAL_FAILURE_)
```

9.24.1.2 ASSERT_PRED2

```
#define ASSERT_PRED2(
    pred,
    v1,
    v2)
```

Wartość:

```
GTEST_PRED2_(pred, v1, v2, GTEST_FATAL_FAILURE_)
```

9.24.1.3 ASSERT_PRED3

```
#define ASSERT_PRED3(
    pred,
    v1,
    v2,
    v3)
```

Wartość:

```
GTEST_PRED3_(pred, v1, v2, v3, GTEST_FATAL_FAILURE_)
```

9.24.1.4 ASSERT_PRED4

```
#define ASSERT_PRED4( pred,  
                    v1,  
                    v2,  
                    v3,  
                    v4)
```

Wartość:

```
GTEST_PRED4_(pred, v1, v2, v3, v4, GTEST_FATAL_FAILURE_)
```

9.24.1.5 ASSERT_PRED5

```
#define ASSERT_PRED5( pred,  
                    v1,  
                    v2,  
                    v3,  
                    v4,  
                    v5)
```

Wartość:

```
GTEST_PRED5_(pred, v1, v2, v3, v4, v5, GTEST_FATAL_FAILURE_)
```

9.24.1.6 ASSERT_PRED_FORMAT1

```
#define ASSERT_PRED_FORMAT1( pred_format,  
                           v1)
```

Wartość:

```
GTEST_PRED_FORMAT1_(pred_format, v1, GTEST_FATAL_FAILURE_)
```

9.24.1.7 ASSERT_PRED_FORMAT2

```
#define ASSERT_PRED_FORMAT2( pred_format,  
                           v1,  
                           v2)
```

Wartość:

```
GTEST_PRED_FORMAT2_(pred_format, v1, v2, GTEST_FATAL_FAILURE_)
```

9.24.1.8 ASSERT_PRED_FORMAT3

```
#define ASSERT_PRED_FORMAT3( pred_format,  
                           v1,  
                           v2,  
                           v3)
```

Wartość:

```
GTEST_PRED_FORMAT3_(pred_format, v1, v2, v3, GTEST_FATAL_FAILURE_)
```

9.24.1.9 ASSERT_PRED_FORMAT4

```
#define ASSERT_PRED_FORMAT4( pred_format, v1, v2, v3, v4)
```

Wartość:

```
GTEST_PRED_FORMAT4_(pred_format, v1, v2, v3, v4, GTEST_FATAL_FAILURE_)
```

9.24.1.10 ASSERT_PRED_FORMAT5

```
#define ASSERT_PRED_FORMAT5( pred_format, v1, v2, v3, v4, v5)
```

Wartość:

```
GTEST_PRED_FORMAT5_(pred_format, v1, v2, v3, v4, v5, GTEST_FATAL_FAILURE_)
```

9.24.1.11 EXPECT_PRED1

```
#define EXPECT_PRED1( pred, v1)
```

Wartość:

```
GTEST_PRED1_(pred, v1, GTEST_NONFATAL_FAILURE_)
```

9.24.1.12 EXPECT_PRED2

```
#define EXPECT_PRED2( pred, v1, v2)
```

Wartość:

```
GTEST_PRED2_(pred, v1, v2, GTEST_NONFATAL_FAILURE_)
```

9.24.1.13 EXPECT_PRED3

```
#define EXPECT_PRED3( pred, v1, v2, v3)
```

Wartość:

```
GTEST_PRED3_(pred, v1, v2, v3, GTEST_NONFATAL_FAILURE_)
```

9.24.1.14 EXPECT_PRED4

```
#define EXPECT_PRED4( pred,  
                    v1,  
                    v2,  
                    v3,  
                    v4)
```

Wartość:

```
GTEST_PRED4_(pred, v1, v2, v3, v4, GTEST_NONFATAL_FAILURE_)
```

9.24.1.15 EXPECT_PRED5

```
#define EXPECT_PRED5( pred,  
                    v1,  
                    v2,  
                    v3,  
                    v4,  
                    v5)
```

Wartość:

```
GTEST_PRED5_(pred, v1, v2, v3, v4, v5, GTEST_NONFATAL_FAILURE_)
```

9.24.1.16 EXPECT_PRED_FORMAT1

```
#define EXPECT_PRED_FORMAT1( pred_format,  
                           v1)
```

Wartość:

```
GTEST_PRED_FORMAT1_(pred_format, v1, GTEST_NONFATAL_FAILURE_)
```

9.24.1.17 EXPECT_PRED_FORMAT2

```
#define EXPECT_PRED_FORMAT2( pred_format,  
                           v1,  
                           v2)
```

Wartość:

```
GTEST_PRED_FORMAT2_(pred_format, v1, v2, GTEST_NONFATAL_FAILURE_)
```

9.24.1.18 EXPECT_PRED_FORMAT3

```
#define EXPECT_PRED_FORMAT3( pred_format,  
                           v1,  
                           v2,  
                           v3)
```

Wartość:

```
GTEST_PRED_FORMAT3_(pred_format, v1, v2, v3, GTEST_NONFATAL_FAILURE_)
```

9.24.1.19 EXPECT_PRED_FORMAT4

```
#define EXPECT_PRED_FORMAT4(
    pred_format,
    v1,
    v2,
    v3,
    v4)
```

Wartość:

```
GTEST_PRED_FORMAT4_(pred_format, v1, v2, v3, v4, GTEST_NONFATAL_FAILURE_)
```

9.24.1.20 EXPECT_PRED_FORMAT5

```
#define EXPECT_PRED_FORMAT5(
    pred_format,
    v1,
    v2,
    v3,
    v4,
    v5)
```

Wartość:

```
GTEST_PRED_FORMAT5_(pred_format, v1, v2, v3, v4, v5, GTEST_NONFATAL_FAILURE_)
```

9.24.1.21 GTEST_ASSERT_

```
#define GTEST_ASSERT_(
    expression,
    on_failure)
```

Wartość:

```
GTEST_AMBIGUOUS_ELSE_BLOCKER_ \
if (const ::testing::AssertionResult gtest_ar = (expression)) \
; \
else \
on_failure(gtest_ar.failure_message())
```

9.24.1.22 GTEST_PRED1_

```
#define GTEST_PRED1_(
    pred,
    v1,
    on_failure)
```

Wartość:

```
GTEST_ASSERT_(::testing::AssertPred1Helper(#pred, \
#v1, \
pred, \
v1), on_failure)
```

9.24.1.23 GTEST_PRED2_

```
#define GTEST_PRED2_(  
    pred,  
    v1,  
    v2,  
    on_failure)
```

Wartość:

```
GTEST_ASSERT_(:testing::AssertPred2Helper(#pred, \  
    #v1, \  
    #v2, \  
    pred, \  
    v1, \  
    v2), on_failure)
```

9.24.1.24 GTEST_PRED3_

```
#define GTEST_PRED3_(  
    pred,  
    v1,  
    v2,  
    v3,  
    on_failure)
```

Wartość:

```
GTEST_ASSERT_(:testing::AssertPred3Helper(#pred, \  
    #v1, \  
    #v2, \  
    #v3, \  
    pred, \  
    v1, \  
    v2, \  
    v3), on_failure)
```

9.24.1.25 GTEST_PRED4_

```
#define GTEST_PRED4_(  
    pred,  
    v1,  
    v2,  
    v3,  
    v4,  
    on_failure)
```

Wartość:

```
GTEST_ASSERT_(:testing::AssertPred4Helper(#pred, \  
    #v1, \  
    #v2, \  
    #v3, \  
    #v4, \  
    pred, \  
    v1, \  
    v2, \  
    v3, \  
    v4), on_failure)
```

9.24.1.26 GTEST_PRED5_

```
#define GTEST_PRED5_(
    pred,
    v1,
    v2,
    v3,
    v4,
    v5,
    on_failure)
```

Wartość:

```
GTEST_ASSERT_(:testing::AssertPred5Helper(#pred, \
    #v1, \
    #v2, \
    #v3, \
    #v4, \
    #v5, \
    pred, \
    v1, \
    v2, \
    v3, \
    v4, \
    v5), on_failure)
```

9.24.1.27 GTEST_PRED_FORMAT1_

```
#define GTEST_PRED_FORMAT1_(
    pred_format,
    v1,
    on_failure)
```

Wartość:

```
GTEST_ASSERT_(pred_format(#v1, v1), \
    on_failure)
```

9.24.1.28 GTEST_PRED_FORMAT2_

```
#define GTEST_PRED_FORMAT2_(
    pred_format,
    v1,
    v2,
    on_failure)
```

Wartość:

```
GTEST_ASSERT_(pred_format(#v1, #v2, v1, v2), \
    on_failure)
```

9.24.1.29 GTEST_PRED_FORMAT3_

```
#define GTEST_PRED_FORMAT3_(
    pred_format,
    v1,
    v2,
    v3,
    on_failure)
```

Wartość:

```
GTEST_ASSERT_(pred_format(#v1, #v2, #v3, v1, v2, v3), \
    on_failure)
```

9.24.1.30 GTEST_PRED_FORMAT4_

```
#define GTEST_PRED_FORMAT4_(
    pred_format,
    v1,
    v2,
    v3,
    v4,
    on_failure)
```

Wartość:

```
GTEST_ASSERT_(pred_format(#v1, #v2, #v3, #v4, v1, v2, v3, v4), \
    on_failure)
```

9.24.1.31 GTEST_PRED_FORMAT5_

```
#define GTEST_PRED_FORMAT5_(
    pred_format,
    v1,
    v2,
    v3,
    v4,
    v5,
    on_failure)
```

Wartość:

```
GTEST_ASSERT_(pred_format(#v1, #v2, #v3, #v4, #v5, v1, v2, v3, v4, v5), \
    on_failure)
```

9.25 gtest_pred_impl.h

[Idź do dokumentacji tego pliku.](#)

```
00001 // Copyright 2006, Google Inc.
00002 // All rights reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //      * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //      * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029
00030 // This file is AUTOMATICALLY GENERATED on 01/02/2018 by command
00031 // 'gen_gtest_pred_impl.py 5'. DO NOT EDIT BY HAND!
00032 //
00033 // Implements a family of generic predicate assertion macros.
```

```

00034 // GOOGLETEST_CM0001 DO NOT DELETE
00035 #ifndef GTEST_INCLUDE_GTEST_GTEST_PRED_IMPL_H_
00036 #define GTEST_INCLUDE_GTEST_GTEST_PRED_IMPL_H_
00037
00038 #include "gtest/gtest.h"
00039
00040 namespace testing {
00041
00042 // This header implements a family of generic predicate assertion
00043 // macros:
00044 //
00045 //   ASSERT_PRED_FORMAT1(pred_format, v1)
00046 //   ASSERT_PRED_FORMAT2(pred_format, v1, v2)
00047 //   ...
00048 //
00049 // where pred_format is a function or functor that takes n (in the
00050 // case of ASSERT_PRED_FORMATn) values and their source expression
00051 // text, and returns a testing::AssertionResult. See the definition
00052 // of ASSERT_EQ in gtest.h for an example.
00053 //
00054 // If you don't care about formatting, you can use the more
00055 // restrictive version:
00056 //
00057 //   ASSERT_PRED1(pred, v1)
00058 //   ASSERT_PRED2(pred, v1, v2)
00059 //   ...
00060 //
00061 // where pred is an n-ary function or functor that returns bool,
00062 // and the values v1, v2, ..., must support the << operator for
00063 // streaming to std::ostream.
00064 //
00065 // We also define the EXPECT_* variations.
00066 //
00067 // For now we only support predicates whose arity is at most 5.
00068 //
00069 // GTEST_ASSERT_ is the basic statement to which all of the assertions
00070 // in this file reduce. Don't use this in your code.
00071
00072 #define GTEST_ASSERT_(expression, on_failure) \
00073     GTEST_AMBIGUOUS_ELSE_BLOCKER_ \
00074     if (const ::testing::AssertionResult gtest_ar = (expression)) \
00075         ; \
00076     else \
00077         on_failure(gtest_ar.failure_message())
00078
00079
00080
00081
00082 // Helper function for implementing {EXPECT|ASSERT}_PRED1. Don't use
00083 // this in your code.
00084 template <typename Pred,
00085             typename T1>
00086 AssertionResult AssertPred1Helper(const char* pred_text,
00087                                     const char* el,
00088                                     Pred pred,
00089                                     const T1& v1) {
00090     if (pred(v1)) return AssertionSuccess();
00091
00092     return AssertionFailure() << pred_text << "("
00093                                     << el << ") evaluates to false, where"
00094                                     << "\n" << el << " evaluates to " << v1;
00095 }
00096
00097 // Internal macro for implementing {EXPECT|ASSERT}_PRED_FORMAT1.
00098 // Don't use this in your code.
00099 #define GTEST_PRED_FORMAT1_(pred_format, v1, on_failure) \
00100     GTEST_ASSERT_(pred_format(#v1, v1), \
00101                 on_failure)
00102
00103 // Internal macro for implementing {EXPECT|ASSERT}_PRED1. Don't use
00104 // this in your code.
00105 #define GTEST_PRED1_(pred, v1, on_failure) \
00106     GTEST_ASSERT_(:testing::AssertPred1Helper(#pred, \
00107                                             #v1, \
00108                                             pred, \
00109                                             v1), on_failure)
00110
00111 // Unary predicate assertion macros.
00112 #define EXPECT_PRED_FORMAT1(pred_format, v1) \
00113     GTEST_PRED_FORMAT1_(pred_format, v1, GTEST_NONFATAL_FAILURE_)
00114 #define EXPECT_PRED1(pred, v1) \
00115     GTEST_PRED1_(pred, v1, GTEST_NONFATAL_FAILURE_)
00116 #define ASSERT_PRED_FORMAT1(pred_format, v1) \
00117     GTEST_PRED_FORMAT1_(pred_format, v1, GTEST_FATAL_FAILURE_)
00118 #define ASSERT_PRED1(pred, v1) \
00119     GTEST_PRED1_(pred, v1, GTEST_FATAL_FAILURE_)
00120

```

```

00121
00122
00123 // Helper function for implementing {EXPECT|ASSERT}_PRED2. Don't use
00124 // this in your code.
00125 template <typename Pred,
00126         typename T1,
00127         typename T2>
00128 AssertionResult AssertPred2Helper(const char* pred_text,
00129                                     const char* e1,
00130                                     const char* e2,
00131                                     Pred pred,
00132                                     const T1& v1,
00133                                     const T2& v2) {
00134     if (pred(v1, v2)) return AssertionSuccess();
00135
00136     return AssertionFailure() << pred_text << "("
00137             << e1 << ", "
00138             << e2 << ") evaluates to false, where"
00139             << "\n" << e1 << " evaluates to " << v1
00140             << "\n" << e2 << " evaluates to " << v2;
00141 }
00142
00143 // Internal macro for implementing {EXPECT|ASSERT}_PRED_FORMAT2.
00144 // Don't use this in your code.
00145 #define GTEST_PRED_FORMAT2_(pred_format, v1, v2, on_failure) \
00146     GTEST_ASSERT_(pred_format(#v1, #v2, v1, v2), \
00147                 on_failure)
00148
00149 // Internal macro for implementing {EXPECT|ASSERT}_PRED2. Don't use
00150 // this in your code.
00151 #define GTEST_PRED2_(pred, v1, v2, on_failure) \
00152     GTEST_ASSERT_(:testing::AssertPred2Helper(#pred, \
00153                                         #v1, \
00154                                         #v2, \
00155                                         pred, \
00156                                         v1, \
00157                                         v2), on_failure)
00158
00159 // Binary predicate assertion macros.
00160 #define EXPECT_PRED_FORMAT2(pred_format, v1, v2) \
00161     GTEST_PRED_FORMAT2_(pred_format, v1, v2, GTEST_NONFATAL_FAILURE_)
00162 #define EXPECT_PRED2(pred, v1, v2) \
00163     GTEST_PRED2_(pred, v1, v2, GTEST_NONFATAL_FAILURE_)
00164 #define ASSERT_PRED_FORMAT2(pred_format, v1, v2) \
00165     GTEST_PRED_FORMAT2_(pred_format, v1, v2, GTEST_FATAL_FAILURE_)
00166 #define ASSERT_PRED2(pred, v1, v2) \
00167     GTEST_PRED2_(pred, v1, v2, GTEST_FATAL_FAILURE_)
00168
00169
00170
00171 // Helper function for implementing {EXPECT|ASSERT}_PRED3. Don't use
00172 // this in your code.
00173 template <typename Pred,
00174         typename T1,
00175         typename T2,
00176         typename T3>
00177 AssertionResult AssertPred3Helper(const char* pred_text,
00178                                     const char* e1,
00179                                     const char* e2,
00180                                     const char* e3,
00181                                     Pred pred,
00182                                     const T1& v1,
00183                                     const T2& v2,
00184                                     const T3& v3) {
00185     if (pred(v1, v2, v3)) return AssertionSuccess();
00186
00187     return AssertionFailure() << pred_text << "("
00188             << e1 << ", "
00189             << e2 << ", "
00190             << e3 << ") evaluates to false, where"
00191             << "\n" << e1 << " evaluates to " << v1
00192             << "\n" << e2 << " evaluates to " << v2
00193             << "\n" << e3 << " evaluates to " << v3;
00194 }
00195
00196 // Internal macro for implementing {EXPECT|ASSERT}_PRED_FORMAT3.
00197 // Don't use this in your code.
00198 #define GTEST_PRED_FORMAT3_(pred_format, v1, v2, v3, on_failure) \
00199     GTEST_ASSERT_(pred_format(#v1, #v2, #v3, v1, v2, v3), \
00200                 on_failure)
00201
00202 // Internal macro for implementing {EXPECT|ASSERT}_PRED3. Don't use
00203 // this in your code.
00204 #define GTEST_PRED3_(pred, v1, v2, v3, on_failure) \
00205     GTEST_ASSERT_(:testing::AssertPred3Helper(#pred, \
00206                                         #v1, \
00207                                         #v2, \

```

```

00208                               #v3, \
00209                               pred, \
00210                               v1, \
00211                               v2, \
00212                               v3), on_failure)
00213
00214 // Ternary predicate assertion macros.
00215 #define EXPECT_PRED_FORMAT3(pred_format, v1, v2, v3) \
00216   GTEST_PRED_FORMAT3_(pred_format, v1, v2, v3, GTEST_NONFATAL_FAILURE_)
00217 #define EXPECT_PRED3(pred, v1, v2, v3) \
00218   GTEST_PRED3_(pred, v1, v2, v3, GTEST_NONFATAL_FAILURE_)
00219 #define ASSERT_PRED_FORMAT3(pred_format, v1, v2, v3) \
00220   GTEST_PRED_FORMAT3_(pred_format, v1, v2, v3, GTEST_FATAL_FAILURE_)
00221 #define ASSERT_PRED3(pred, v1, v2, v3) \
00222   GTEST_PRED3_(pred, v1, v2, v3, GTEST_FATAL_FAILURE_)
00223
00224
00225
00226 // Helper function for implementing {EXPECT|ASSERT}_PRED4. Don't use
00227 // this in your code.
00228 template <typename Pred,
00229           typename T1,
00230           typename T2,
00231           typename T3,
00232           typename T4>
00233 AssertionResult AssertPred4Helper(const char* pred_text,
00234                                     const char* e1,
00235                                     const char* e2,
00236                                     const char* e3,
00237                                     const char* e4,
00238                                     Pred pred,
00239                                     const T1& v1,
00240                                     const T2& v2,
00241                                     const T3& v3,
00242                                     const T4& v4) {
00243   if (pred(v1, v2, v3, v4)) return AssertionSuccess();
00244
00245   return AssertionFailure() << pred_text << "("
00246     << e1 << ", "
00247     << e2 << ", "
00248     << e3 << ", "
00249     << e4 << ") evaluates to false, where"
00250     << "\n" << e1 << " evaluates to " << v1
00251     << "\n" << e2 << " evaluates to " << v2
00252     << "\n" << e3 << " evaluates to " << v3
00253     << "\n" << e4 << " evaluates to " << v4;
00254 }
00255
00256 // Internal macro for implementing {EXPECT|ASSERT}_PRED_FORMAT4.
00257 // Don't use this in your code.
00258 #define GTEST_PRED_FORMAT4_(pred_format, v1, v2, v3, v4, on_failure) \
00259   GTEST_ASSERT_(pred_format(#v1, #v2, #v3, #v4, v1, v2, v3, v4), \
00260                 on_failure)
00261
00262 // Internal macro for implementing {EXPECT|ASSERT}_PRED4. Don't use
00263 // this in your code.
00264 #define GTEST_PRED4_(pred, v1, v2, v3, v4, on_failure) \
00265   GTEST_ASSERT_(<:testing::AssertPred4Helper(pred, \
00266                                             #v1, \
00267                                             #v2, \
00268                                             #v3, \
00269                                             #v4, \
00270                                             pred, \
00271                                             v1, \
00272                                             v2, \
00273                                             v3, \
00274                                             v4), on_failure)
00275
00276 // 4-ary predicate assertion macros.
00277 #define EXPECT_PRED_FORMAT4(pred_format, v1, v2, v3, v4) \
00278   GTEST_PRED_FORMAT4_(pred_format, v1, v2, v3, v4, GTEST_NONFATAL_FAILURE_)
00279 #define EXPECT_PRED4(pred, v1, v2, v3, v4) \
00280   GTEST_PRED4_(pred, v1, v2, v3, v4, GTEST_NONFATAL_FAILURE_)
00281 #define ASSERT_PRED_FORMAT4(pred_format, v1, v2, v3, v4) \
00282   GTEST_PRED_FORMAT4_(pred_format, v1, v2, v3, v4, GTEST_FATAL_FAILURE_)
00283 #define ASSERT_PRED4(pred, v1, v2, v3, v4) \
00284   GTEST_PRED4_(pred, v1, v2, v3, v4, GTEST_FATAL_FAILURE_)
00285
00286
00287
00288 // Helper function for implementing {EXPECT|ASSERT}_PRED5. Don't use
00289 // this in your code.
00290 template <typename Pred,
00291           typename T1,
00292           typename T2,
00293           typename T3,
00294           typename T4,

```

```

00295     typename T5>
00296 AssertionResult AssertPred5Helper(const char* pred_text,
00297                                     const char* e1,
00298                                     const char* e2,
00299                                     const char* e3,
00300                                     const char* e4,
00301                                     const char* e5,
00302                                     Pred pred,
00303                                     const T1& v1,
00304                                     const T2& v2,
00305                                     const T3& v3,
00306                                     const T4& v4,
00307                                     const T5& v5) {
00308     if (pred(v1, v2, v3, v4, v5)) return AssertionSuccess();
00309
00310     return AssertionFailure() << pred_text << "("
00311         << e1 << ", "
00312         << e2 << ", "
00313         << e3 << ", "
00314         << e4 << ", "
00315         << e5 << ") evaluates to false, where"
00316         << "\n" << e1 << " evaluates to " << v1
00317         << "\n" << e2 << " evaluates to " << v2
00318         << "\n" << e3 << " evaluates to " << v3
00319         << "\n" << e4 << " evaluates to " << v4
00320         << "\n" << e5 << " evaluates to " << v5;
00321 }
00322
00323 // Internal macro for implementing {EXPECT|ASSERT}_PRED_FORMAT5.
00324 // Don't use this in your code.
00325 #define GTEST_PRED_FORMAT5_(pred_format, v1, v2, v3, v4, v5, on_failure)\ \
00326     GTEST_ASSERT_((pred_format #v1, #v2, #v3, #v4, #v5, v1, v2, v3, v4, v5), \ \
00327     on_failure)
00328
00329 // Internal macro for implementing {EXPECT|ASSERT}_PRED5. Don't use
00330 // this in your code.
00331 #define GTEST_PRED5_(pred, v1, v2, v3, v4, v5, on_failure)\ \
00332     GTEST_ASSERT_(:testing::AssertPred5Helper(#pred, \
00333         #v1, \
00334         #v2, \
00335         #v3, \
00336         #v4, \
00337         #v5, \
00338         pred, \
00339         v1, \
00340         v2, \
00341         v3, \
00342         v4, \
00343         v5), on_failure)
00344
00345 // 5-ary predicate assertion macros.
00346 #define EXPECT_PRED_FORMAT5(pred_format, v1, v2, v3, v4, v5) \
00347     GTEST_PRED_FORMAT5_(pred_format, v1, v2, v3, v4, v5, GTEST_NONFATAL_FAILURE_)
00348 #define EXPECT_PRED5(pred, v1, v2, v3, v4, v5) \
00349     GTEST_PRED5_(pred, v1, v2, v3, v4, v5, GTEST_NONFATAL_FAILURE_)
00350 #define ASSERT_PRED_FORMAT5(pred_format, v1, v2, v3, v4, v5) \
00351     GTEST_PRED_FORMAT5_(pred_format, v1, v2, v3, v4, v5, GTEST_FATAL_FAILURE_)
00352 #define ASSERT_PRED5(pred, v1, v2, v3, v4, v5) \
00353     GTEST_PRED5_(pred, v1, v2, v3, v4, v5, GTEST_FATAL_FAILURE_)
00354
00355
00356
00357 } // namespace testing
00358
00359 #endif // GTEST_INCLUDE_GTEST_GTEST_PRED_IMPL_H_

```

9.26 Dokumentacja pliku **packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest_prod.h**

Definicje

- `#define FRIEND_TEST(test_case_name, test_name)`

9.26.1 Dokumentacja definicji

9.26.1.1 FRIEND_TEST

```
#define FRIEND_TEST(
    test_case_name,
    test_name)
```

Wartość:

```
friend class test_case_name##_##test_name##_Test
```

9.27 gtest_prod.h

[Idź do dokumentacji tego pliku.](#)

```
00001 // Copyright 2006, Google Inc.
00002 // All rights reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //      * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //      * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029
00030 //
00031 // Google C++ Testing and Mocking Framework definitions useful in production code.
00032 // GOOGLETEST_CM0003 DO NOT DELETE
00033
00034 #ifndef GTEST_INCLUDE_GTEST_GTEST_PROD_H_
00035 #define GTEST_INCLUDE_GTEST_GTEST_PROD_H_
00036
00037 // When you need to test the private or protected members of a class,
00038 // use the FRIEND_TEST macro to declare your tests as friends of the
00039 // class. For example:
00040 //
00041 // class MyClass {
00042 //   private:
00043 //     void PrivateMethod();
00044 //     FRIEND_TEST(MyClassTest, PrivateMethodWorks);
00045 // };
00046 //
00047 // class MyClassTest : public testing::Test {
00048 //   // ...
00049 // };
00050 //
00051 // TEST_F(MyClassTest, PrivateMethodWorks) {
00052 //   // Can call MyClass::PrivateMethod() here.
00053 // }
00054 //
00055 // Note: The test class must be in the same namespace as the class being tested.
00056 // For example, putting MyClassTest in an anonymous namespace will not work.
00057
00058 #define FRIEND_TEST(test_case_name, test_name) \
00059 friend class test_case_name##_##test_name##_Test
00060
00061 #endif // GTEST_INCLUDE_GTEST_GTEST_PROD_H_
```

9.28 Dokumentacja pliku packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/custom/gtest-port.h

9.29 gtest-port.h

Idź do dokumentacji tego pliku.

```
00001 // Copyright 2015, Google Inc.
00002 // All rights reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //     * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //     * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //     * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 // Injection point for custom user configurations. See README for details
00031 //
00032 // ** Custom implementation starts here **
00033
00034 #ifndef GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_PORT_H_
00035 #define GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_PORT_H_
00036
00037 #endif // GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_PORT_H_
```

9.30 Dokumentacja pliku packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port.h

```
#include <ctype.h>
#include <stddef.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <string>
#include <algorithm>
#include <iostream>
#include <sstream>
#include <utility>
#include <vector>
#include "gtest/internal/gtest-port-arch.h"
```

```
#include "gtest/internal/custom/gtest-port.h"
#include <unistd.h>
#include <strings.h>
#include <regex.h>
#include <typeinfo>
#include "gtest/internal/gtest-tuple.h"
```

Komponenty

- struct `testing::internal::CompileAssert< bool >`
- struct `testing::internal::StaticAssertTypeEqHelper< T, T >`
- struct `testing::internal::IsSame< T, U >`
- struct `testing::internal::IsSame< T, T >`
- class `testing::internal::scoped_ptr< T >`
- class `testing::internal::RE`
- class `testing::internal::GTestLog`
- struct `testing::internal::AddReference< T >`
- struct `testing::internal::AddReference< T & >`
- struct `testing::internal::ConstRef< T >`
- struct `testing::internal::ConstRef< T & >`
- struct `testing::internal::RvalueRef< T >`
- class `testing::internal::Mutex`
- class `testing::internal::GTestMutexLock`
- class `testing::internal::ThreadLocal< T >`
- struct `testing::internal::bool_constant< bool_value >`
- struct `testing::internal::is_same< T, U >`
- struct `testing::internal::is_same< T, T >`
- struct `testing::internal::is_pointer< T >`
- struct `testing::internal::is_pointer< T * >`
- struct `testing::internal::IteratorTraits< Iterator >`
- struct `testing::internal::IteratorTraits< T * >`
- struct `testing::internal::IteratorTraits< const T * >`
- class `testing::internal::TypeWithSize< size >`
- class `testing::internal::TypeWithSize< 4 >`
- class `testing::internal::TypeWithSize< 8 >`

Przestrzenie nazw

- namespace `testing`
- namespace `testing::internal`
- namespace `testing::internal::posix`

Definicje

- `#define GTEST_DEV_EMAIL_ "googletestframework@@googlegroups.com"`
- `#define GTEST_FLAG_PREFIX_ "gtest_"`
- `#define GTEST_FLAG_PREFIX_DASH_ "gtest-"`
- `#define GTEST_FLAG_PREFIX_UPPER_ "GTEST_"`
- `#define GTEST_NAME_ "Google Test"`
- `#define GTEST_PROJECT_URL_ "https://github.com/google/googletest/"`
- `#define GTEST_INIT_GOOGLE_TEST_NAME_ "testing::InitGoogleTest"`

- #define GTEST_DISABLE_MSC_WARNINGS_PUSH_(warnings)
- #define GTEST_DISABLE_MSC_WARNINGS_POP_()
- #define GTEST_DISABLE_MSC_DEPRECATED_PUSH_()
- #define GTEST_DISABLE_MSC_DEPRECATED_POP_()
- #define GTEST_LANG_CXX11 0
- #define GTEST_HAS_POSIX_RE (!GTEST_OS_WINDOWS)
- #define GTEST_USES_POSIX_RE 1
- #define GTEST_HAS_EXCEPTIONS 0
- #define GTEST_HAS_STD_STRING 1
- #define GTEST_HAS_GLOBAL_STRING 0
- #define GTEST_HAS_STD_WSTRING (!GTEST_OS_LINUX_ANDROID || GTEST_OS_CYGWIN || GTEST_OS_SOLARIS))
- #define GTEST_HAS_GLOBAL_WSTRING (GTEST_HAS_STD_WSTRING && GTEST_HAS_GLOBAL_STRING)
- #define GTEST_HAS_RTTI 1
- #define GTEST_HAS_PTHREAD
- #define GTEST_HAS_TR1_TUPLE 1
- #define GTEST_USE_OWN_TR1_TUPLE 1
- #define GTEST_TUPLE_NAMESPACE_ ::std::tr1
- #define GTEST_HAS_CLONE 0
- #define GTEST_HAS_STREAM_REDIRECTION 1
- #define GTEST_HAS_COMBINE 1
- #define GTEST_WIDE_STRINGUSES_UTF16_ (GTEST_OS_WINDOWS || GTEST_OS_CYGWIN || GTEST_OS_SYMBIAN || GTEST_OS_AIX)
- #define GTEST_AMBIGUOUS_ELSE_BLOCKER_ switch (0) case 0: default:
- #define GTEST_ATTRIBUTE_UNUSED_
- #define GTEST_CXX11_EQUALS_DELETE_
- #define GTEST_ATTRIBUTE_PRINTF_(string_index, first_to_check)
- #define GTEST_DISALLOW_ASSIGN_(type)
- #define GTEST_DISALLOW_COPY_AND_ASSIGN_(type)
- #define GTEST_MUST_USE_RESULT_
- #define GTEST_INTENTIONAL_CONST_COND_PUSH_()
- #define GTEST_INTENTIONAL_CONST_COND_POP_()
- #define GTEST_HAS_SEH 0
- #define GTEST_IS_THREADSAFE
- #define GTEST_API_
- #define GTEST_DEFAULT_DEATH_TEST_STYLE "fast"
- #define GTEST_NO_INLINE_
- #define GTEST_HAS_CXXABI_H_ 0
- #define GTEST_ATTRIBUTE_NO_SANITIZE_MEMORY_
- #define GTEST_ATTRIBUTE_NO_SANITIZE_ADDRESS_
- #define GTEST_ATTRIBUTE_NO_SANITIZE_THREAD_
- #define GTEST_COMPILE_ASSERT_(expr, msg)
- #define GTEST_ARRAY_SIZE_(array)
- #define GTEST_LOG_(severity)
- #define GTEST_CHECK_(condition)
- #define GTEST_CHECK_POSIX_SUCCESS_(posix_call)
- #define GTEST_ADD_REFERENCE_(T)
- #define GTEST_REFERENCE_TO_CONST_(T)
- #define GTEST_DECLARE_STATIC_MUTEX_(mutex)
- #define GTEST_DEFINE_STATIC_MUTEX_(mutex)
- #define GTEST_CAN_COMPARE_NULL 1
- #define GTEST_PATH_SEP_ "/"
- #define GTEST_HAS_ALT_PATH_SEP_ 0
- #define GTEST_SNPRINTF_ snprintf
- #define GTEST_FLAG(name)

- `#define GTEST_USE_OWN_FLAGFILE_FLAG_ 1`
- `#define GTEST_FLAG_SAVER_ ::testing::internal::GTestFlagSaver`
- `#define GTEST_DECLARE_bool_(name)`
- `#define GTEST_DECLARE_int32_(name)`
- `#define GTEST_DECLARE_string_(name)`
- `#define GTEST_DEFINE_bool_(name, default_val, doc)`
- `#define GTEST_DEFINE_int32_(name, default_val, doc)`
- `#define GTEST_DEFINE_string_(name, default_val, doc)`
- `#define GTEST_EXCLUSIVE_LOCK_REQUIRED_(locks)`
- `#define GTEST_LOCK_EXCLUDED_(locks)`

Definicje typów

- `typedef ::std::string testing::internal::string`
- `typedef ::std::wstring testing::internal::wstring`
- `typedef GTestMutexLock testing::internal::MutexLock`
- `typedef bool_constant< false > testing::internal::false_type`
- `typedef bool_constant< true > testing::internal::true_type`
- `typedef long long testing::internal::BiggestInt`
- `typedef struct stat testing::internal::posix::StatStruct`
- `typedef TypeWithSize< 4 >::Int testing::internal::Int32`
- `typedef TypeWithSize< 4 >::UInt testing::internal::UInt32`
- `typedef TypeWithSize< 8 >::Int testing::internal::Int64`
- `typedef TypeWithSize< 8 >::UInt testing::internal::UInt64`
- `typedef TypeWithSize< 8 >::Int testing::internal::TimeInMillis`

Wyliczenia

- enum `testing::internal::GTestLogSeverity { testing::internal::GTEST_INFO , testing::internal::GTEST_WARNING , testing::internal::GTEST_ERROR , testing::internal::GTEST_FATAL }`

Funkcje

- `GTEST_API_ bool testing::internal::IsTrue (bool condition)`
- `GTEST_API_ ::std::string testing::internal::FormatFileLocation (const char *file, int line)`
- `GTEST_API_ ::std::string testing::internal::FormatCompilerIndependentFileLocation (const char *file, int line)`
- `void testing::internal::LogToStderr ()`
- `void testing::internal::FlushInfoLog ()`
- template<typename T>
 `const T & testing::internal::move (const T &t)`
- template<typename T>
 `testing::internal::GTEST_ADD_REFERENCE_ (T) forward(GTEST_ADD_REFERENCE_(T) t)`
- template<typename To>
 `To testing::internal::ImplicitCast_ (To x)`
- template<typename To, typename From>
 `To testing::internal::DownCast_ (From *f)`
- template<class Derived, class Base>
 `Derived * testing::internal::CheckedDowncastToActualType (Base *base)`
- `GTEST_API_ void testing::internal::CaptureStdout ()`
- `GTEST_API_ std::string testing::internal::GetCapturedStdout ()`
- `GTEST_API_ void testing::internal::CaptureStderr ()`
- `GTEST_API_ std::string testing::internal::GetCapturedStderr ()`

- `GTEST_API_ size_t testing::internal::GetFileSize (FILE *file)`
- `GTEST_API_ std::string testing::internal::ReadEntireFile (FILE *file)`
- `GTEST_API_ std::vector< std::string > testing::internal::GetArgs ()`
- `GTEST_API_ size_t testing::internal::GetThreadCount ()`
- `bool testing::internal::IsAlpha (char ch)`
- `bool testing::internal::IsAlNum (char ch)`
- `bool testing::internal::IsDigit (char ch)`
- `bool testing::internal::IsLower (char ch)`
- `bool testing::internal::IsSpace (char ch)`
- `bool testing::internal::IsUpper (char ch)`
- `bool testing::internal::IsXDigit (char ch)`
- `bool testing::internal::IsXDigit (wchar_t ch)`
- `char testing::internal::ToLower (char ch)`
- `char testing::internal::ToUpper (char ch)`
- `std::string testing::internal::StripTrailingSpaces (std::string str)`
- `int testing::internal::posix::FileNo (FILE *file)`
- `int testing::internal::posix::IsATTY (int fd)`
- `int testing::internal::posix::Stat (const char *path, StatStruct *buf)`
- `int testing::internal::posix::StrCaseCmp (const char *s1, const char *s2)`
- `char * testing::internal::posix::StrDup (const char *src)`
- `int testing::internal::posix::RmDir (const char *dir)`
- `bool testing::internal::posix::IsDir (const StatStruct &st)`
- `const char * testing::internal::posix::StrNCpy (char *dest, const char *src, size_t n)`
- `int testing::internal::posix::ChDir (const char *dir)`
- `FILE * testing::internal::posix::FOpen (const char *path, const char *mode)`
- `FILE * testing::internal::posix::FReopen (const char *path, const char *mode, FILE *stream)`
- `FILE * testing::internal::posix::FDOpen (int fd, const char *mode)`
- `int testing::internal::posix::FClose (FILE *fp)`
- `int testing::internal::posix::Read (int fd, void *buf, unsigned int count)`
- `int testing::internal::posix::Write (int fd, const void *buf, unsigned int count)`
- `int testing::internal::posix::Close (int fd)`
- `const char * testing::internal::posix::StrError (int errnum)`
- `const char * testing::internal::posix::GetEnv (const char *name)`
- `void testing::internal::posix::Abort ()`
- `bool testing::internal::ParseInt32 (const Message &src_text, const char *str, Int32 *value)`
- `bool testing::internal::BoolFromGTestEnv (const char *flag, bool default_val)`
- `GTEST_API_ Int32 testing::internal::Int32FromGTestEnv (const char *flag, Int32 default_val)`
- `std::string testing::internal::OutputFlagAlsoCheckEnvVar ()`
- `const char * testing::internal::StringFromGTestEnv (const char *flag, const char *default_val)`

Zmienne

- `template<bool bool_value>`
`const bool testing::internal::bool_constant< bool_value >::value`
- `const BiggestInt testing::internal::kMaxBiggestInt`

9.30.1 Dokumentacja definicji

9.30.1.1 GTEST_ADD_REFERENCE_

```
#define GTEST_ADD_REFERENCE_(
    T)
```

Wartość:

```
typename ::testing::internal::AddReference<T>::type
```

9.30.1.2 GTEST_AMBIGUOUS_ELSE_BLOCKER_

```
#define GTEST_AMBIGUOUS_ELSE_BLOCKER_ switch (0) case 0: default:
```

9.30.1.3 GTEST_API_

```
#define GTEST_API_
```

9.30.1.4 GTEST_ARRAY_SIZE_

```
#define GTEST_ARRAY_SIZE_(
    array)
```

Wartość:

```
(sizeof(array) / sizeof(array[0]))
```

9.30.1.5 GTEST_ATTRIBUTE_NO_SANITIZE_ADDRESS_

```
#define GTEST_ATTRIBUTE_NO_SANITIZE_ADDRESS_
```

9.30.1.6 GTEST_ATTRIBUTE_NO_SANITIZE_MEMORY_

```
#define GTEST_ATTRIBUTE_NO_SANITIZE_MEMORY_
```

9.30.1.7 GTEST_ATTRIBUTE_NO_SANITIZE_THREAD_

```
#define GTEST_ATTRIBUTE_NO_SANITIZE_THREAD_
```

9.30.1.8 GTEST_ATTRIBUTE_PRINTF_

```
#define GTEST_ATTRIBUTE_PRINTF_(
    string_index,
    first_to_check)
```

9.30.1.9 GTEST_ATTRIBUTE_UNUSED_

```
#define GTEST_ATTRIBUTE_UNUSED_
```

9.30.1.10 GTEST_CAN_COMPARE_NULL

```
#define GTEST_CAN_COMPARE_NULL 1
```

9.30.1.11 GTEST_CHECK_

```
#define GTEST_CHECK_(  
    condition)
```

Wartość:

```
GTEST_AMBIGUOUS_ELSE_BLOCKER_ \  
if (::testing::internal::IsTrue(condition)) \  
; \  
else \  
    GTEST_LOG_(FATAL) << "Condition " #condition " failed. "
```

9.30.1.12 GTEST_CHECK_POSIX_SUCCESS_

```
#define GTEST_CHECK_POSIX_SUCCESS_(  
    posix_call)
```

Wartość:

```
if (const int gtest_error = (posix_call)) \  
    GTEST_LOG_(FATAL) << "#posix_call failed with error " \  
    << gtest_error
```

9.30.1.13 GTEST_COMPILE_ASSERT_

```
#define GTEST_COMPILE_ASSERT_(  
    expr,  
    msg)
```

Wartość:

```
typedef ::testing::internal::CompileAssert<(static_cast<bool>(expr))> \  
msg[static_cast<bool>(expr) ? 1 : -1] GTEST_ATTRIBUTE_UNUSED_
```

9.30.1.14 GTEST_CXX11_EQUALS_DELETE_

```
#define GTEST_CXX11_EQUALS_DELETE_
```

9.30.1.15 GTEST_DECLARE_bool_

```
#define GTEST_DECLARE_bool_(  
    name)
```

Wartość:

```
GTEST_API_ extern bool GTEST_FLAG(name)
```

9.30.1.16 GTEST_DECLARE_int32_

```
#define GTEST_DECLARE_int32_(  
    name)
```

Wartość:

```
GTEST_API_ extern ::testing::internal::Int32 GTEST_FLAG(name)
```

9.30.1.17 GTEST_DECLARE_STATIC_MUTEX_

```
#define GTEST_DECLARE_STATIC_MUTEX_(
    mutex)
```

Wartość:

```
extern ::testing::internal::Mutex mutex
```

9.30.1.18 GTEST_DECLARE_string_

```
#define GTEST_DECLARE_string_(
    name)
```

Wartość:

```
GTEST_API_ extern ::std::string GTEST_FLAG(name)
```

9.30.1.19 GTEST_DEFAULT_DEATH_TEST_STYLE

```
#define GTEST_DEFAULT_DEATH_TEST_STYLE "fast"
```

9.30.1.20 GTEST_DEFINE_bool_

```
#define GTEST_DEFINE_bool_(
    name,
    default_val,
    doc)
```

Wartość:

```
GTEST_API_ bool GTEST_FLAG(name) = (default_val)
```

9.30.1.21 GTEST_DEFINE_int32_

```
#define GTEST_DEFINE_int32_(
    name,
    default_val,
    doc)
```

Wartość:

```
GTEST_API_ ::testing::internal::Int32 GTEST_FLAG(name) = (default_val)
```

9.30.1.22 GTEST_DEFINE_STATIC_MUTEX_

```
#define GTEST_DEFINE_STATIC_MUTEX_(
    mutex)
```

Wartość:

```
::testing::internal::Mutex mutex
```

9.30.1.23 GTEST_DEFINE_string_

```
#define GTEST_DEFINE_string_(  
    name,  
    default_val,  
    doc)
```

Wartość:

```
GTEST_API_ ::std::string GTEST_FLAG(name) = (default_val)
```

9.30.1.24 GTEST_DEV_EMAIL_

```
#define GTEST_DEV_EMAIL_ "googletestframework@googlegroups.com"
```

9.30.1.25 GTEST_DISABLE_MSC_DEPRECATED_POP_

```
#define GTEST_DISABLE_MSC_DEPRECATED_POP_()
```

Wartość:

```
GTEST_DISABLE_MSC_WARNINGS_POP_()
```

9.30.1.26 GTEST_DISABLE_MSC_DEPRECATED_PUSH_

```
#define GTEST_DISABLE_MSC_DEPRECATED_PUSH_()
```

Wartość:

```
GTEST_DISABLE_MSC_WARNINGS_PUSH_(4996)
```

9.30.1.27 GTEST_DISABLE_MSC_WARNINGS_POP_

```
#define GTEST_DISABLE_MSC_WARNINGS_POP_()
```

9.30.1.28 GTEST_DISABLE_MSC_WARNINGS_PUSH_

```
#define GTEST_DISABLE_MSC_WARNINGS_PUSH_(  
    warnings)
```

9.30.1.29 GTEST_DISALLOW_ASSIGN_

```
#define GTEST_DISALLOW_ASSIGN_(  
    type)
```

Wartość:

```
void operator=(type const &) GTEST_CXX11_EQUALS_DELETE_
```

9.30.1.30 GTEST_DISALLOW_COPY_AND_ASSIGN_

```
#define GTEST_DISALLOW_COPY_AND_ASSIGN_(  
    type)  
    type(type const &) GTEST_CXX11_EQUALS_DELETE_; \  
    GTEST_DISALLOW_ASSIGN_(type)
```

Wartość:

```
type(type const &) GTEST_CXX11_EQUALS_DELETE_; \  
GTEST_DISALLOW_ASSIGN_(type)
```

9.30.1.31 GTEST_EXCLUSIVE_LOCK_REQUIRED_

```
#define GTEST_EXCLUSIVE_LOCK_REQUIRED_(  
    locks)
```

9.30.1.32 GTEST_FLAG

```
#define GTEST_FLAG(  
    name)
```

Wartość:

```
FLAGS_gtest_##name
```

9.30.1.33 GTEST_FLAG_PREFIX_

```
#define GTEST_FLAG_PREFIX_ "gtest_"
```

9.30.1.34 GTEST_FLAG_PREFIX_DASH_

```
#define GTEST_FLAG_PREFIX_DASH_ "gtest-"
```

9.30.1.35 GTEST_FLAG_PREFIX_UPPER_

```
#define GTEST_FLAG_PREFIX_UPPER_ "GTEST_"
```

9.30.1.36 GTEST_FLAG_SAVER_

```
#define GTEST_FLAG_SAVER_ ::testing::internal::GTestFlagSaver
```

9.30.1.37 GTEST_HAS_ALT_PATH_SEP_

```
#define GTEST_HAS_ALT_PATH_SEP_ 0
```

9.30.1.38 GTEST_HAS_CLONE

```
#define GTEST_HAS_CLONE 0
```

9.30.1.39 GTEST_HAS_COMBINE

```
#define GTEST_HAS_COMBINE 1
```

9.30.1.40 GTEST_HAS_CXXABI_H_

```
#define GTEST_HAS_CXXABI_H_ 0
```

9.30.1.41 GTEST_HAS_EXCEPTIONS

```
#define GTEST_HAS_EXCEPTIONS 0
```

9.30.1.42 GTEST_HAS_GLOBAL_STRING

```
#define GTEST_HAS_GLOBAL_STRING 0
```

9.30.1.43 GTEST_HAS_GLOBAL_WSTRING

```
#define GTEST_HAS_GLOBAL_WSTRING (GTEST_HAS_STD_WSTRING && GTEST_HAS_GLOBAL_STRING)
```

9.30.1.44 GTEST_HAS_POSIX_REGEX

```
#define GTEST_HAS_POSIX_REGEX (!GTEST_OS_WINDOWS)
```

9.30.1.45 GTEST_HAS_PTHREAD

```
#define GTEST_HAS_PTHREAD
```

Wartość:

```
(GTEST_OS_LINUX || GTEST_OS_MAC || GTEST_OS_HPUX || GTEST_OS_QNX || \  
GTEST_OS_FREEBSD || GTEST_OS_NACL || GTEST_OS_NETBSD || GTEST_OS_FUCHSIA)
```

9.30.1.46 GTEST_HAS_RTTI

```
#define GTEST_HAS_RTTI 1
```

9.30.1.47 GTEST_HAS_SEH

```
#define GTEST_HAS_SEH 0
```

9.30.1.48 GTEST_HAS_STD_STRING

```
#define GTEST_HAS_STD_STRING 1
```

9.30.1.49 GTEST_HAS_STD_WSTRING

```
#define GTEST_HAS_STD_WSTRING  (! (GTEST_OS_LINUX_ANDROID || GTEST_OS_CYGWIN || GTEST_OS_SOLARIS))
```

9.30.1.50 GTEST_HAS_STREAM_REDIRECTION

```
#define GTEST_HAS_STREAM_REDIRECTION 1
```

9.30.1.51 GTEST_HAS_TR1_TUPLE

```
#define GTEST_HAS_TR1_TUPLE 1
```

9.30.1.52 GTEST_INIT_GOOGLE_TEST_NAME_

```
#define GTEST_INIT_GOOGLE_TEST_NAME_ "testing::InitGoogleTest"
```

9.30.1.53 GTEST_INTENTIONAL_CONST_COND_POP_

```
#define GTEST_INTENTIONAL_CONST_COND_POP_()
```

Wartość:

```
GTEST_DISABLE_MSC_WARNINGS_POP_()
```

9.30.1.54 GTEST_INTENTIONAL_CONST_COND_PUSH_

```
#define GTEST_INTENTIONAL_CONST_COND_PUSH_()
```

Wartość:

```
GTEST_DISABLE_MSC_WARNINGS_PUSH_(4127)
```

9.30.1.55 GTEST_IS_THREADSAFE

```
#define GTEST_IS_THREADSAFE
```

Wartość:

```
(GTEST_HAS_MUTEX_AND_THREAD_LOCAL_ \
 || (GTEST_OS_WINDOWS && !GTEST_OS_WINDOWS_PHONE && !GTEST_OS_WINDOWS_RT) \
 || GTEST_HAS_PTHREAD)
```

9.30.1.56 GTEST_LANG_CXX11

```
#define GTEST_LANG_CXX11 0
```

9.30.1.57 GTEST_LOCK_EXCLUDED_

```
#define GTEST_LOCK_EXCLUDED_(
    locks)
```

9.30.1.58 GTEST_LOG_

```
#define GTEST_LOG_(
    severity)
```

Wartość:

```
::testing::internal::GTestLog(::testing::internal::GTEST_##severity, \
    __FILE__, __LINE__).GetStream()
```

9.30.1.59 GTEST_MUST_USE_RESULT_

```
#define GTEST_MUST_USE_RESULT_
```

9.30.1.60 GTEST_NAME_

```
#define GTEST_NAME_ "Google Test"
```

9.30.1.61 GTEST_NO_INLINE_

```
#define GTEST_NO_INLINE_
```

9.30.1.62 GTEST_PATH_SEP_

```
#define GTEST_PATH_SEP_ "/"
```

9.30.1.63 GTEST_PROJECT_URL_

```
#define GTEST_PROJECT_URL_ "https://github.com/google/googletest/"
```

9.30.1.64 GTEST_REFERENCE_TO_CONST_

```
#define GTEST_REFERENCE_TO_CONST_(
    T)
```

Wartość:

```
typename ::testing::internal::ConstRef<T>::type
```

9.30.1.65 GTEST_SNPRINTF_

```
#define GTEST_SNPRINTF_ snprintf
```

9.30.1.66 GTEST_TUPLE_NAMESPACE_

```
#define GTEST_TUPLE_NAMESPACE_ ::std::tr1
```

9.30.1.67 GTEST_USE_OWN_FLAGFILE_FLAG_

```
#define GTEST_USE_OWN_FLAGFILE_FLAG_ 1
```

9.30.1.68 GTEST_USE_OWN_TR1_TUPLE

```
#define GTEST_USE_OWN_TR1_TUPLE 1
```

9.30.1.69 GTESTUSES_POSIX_RE

```
#define GTESTUSES_POSIX_RE 1
```

9.30.1.70 GTEST_WIDE_STRING_USES_UTF16_

```
#define GTEST_WIDE_STRING_USES_UTF16_ (GTEST_OS_WINDOWS || GTEST_OS_CYGWIN || GTEST_OS_SYMBIAN  
|| GTEST_OS_AIX)
```

9.31 gtest-port.h

[Idź do dokumentacji tego pliku.](#)

```
00001 // Copyright 2005, Google Inc.  
00002 // All rights reserved.  
00003 //  
00004 // Redistribution and use in source and binary forms, with or without  
00005 // modification, are permitted provided that the following conditions are  
00006 // met:  
00007 //  
00008 //      * Redistributions of source code must retain the above copyright  
00009 // notice, this list of conditions and the following disclaimer.  
00010 //      * Redistributions in binary form must reproduce the above  
00011 // copyright notice, this list of conditions and the following disclaimer  
00012 // in the documentation and/or other materials provided with the  
00013 // distribution.  
00014 //      * Neither the name of Google Inc. nor the names of its  
00015 // contributors may be used to endorse or promote products derived from  
00016 // this software without specific prior written permission.  
00017 //  
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS  
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT  
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR  
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT  
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,  
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT  
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,  
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY  
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE  
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
00029 //  
00030 // Low-level types and utilities for porting Google Test to various  
00031 // platforms. All macros ending with _ and symbols defined in an  
00032 // internal namespace are subject to change without notice. Code  
00033 // outside Google Test MUST NOT USE THEM DIRECTLY. Macros that don't  
00034 // end with _ are part of Google Test's public API and can be used by  
00035 // code outside Google Test.  
00036 //
```

```
00037 // This file is fundamental to Google Test. All other Google Test source
00038 // files are expected to #include this. Therefore, it cannot #include
00039 // any other Google Test header.
00040
00041 // GOOGLETTEST_CM0001 DO NOT DELETE
00042
00043 #ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PORT_H_
00044 #define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PORT_H_
00045
00046 // Environment-describing macros
00047 // -----
00048 //
00049 // Google Test can be used in many different environments. Macros in
00050 // this section tell Google Test what kind of environment it is being
00051 // used in, such that Google Test can provide environment-specific
00052 // features and implementations.
00053 //
00054 // Google Test tries to automatically detect the properties of its
00055 // environment, so users usually don't need to worry about these
00056 // macros. However, the automatic detection is not perfect.
00057 // Sometimes it's necessary for a user to define some of the following
00058 // macros in the build script to override Google Test's decisions.
00059 //
00060 // If the user doesn't define a macro in the list, Google Test will
00061 // provide a default definition. After this header is #included, all
00062 // macros in this list will be defined to either 1 or 0.
00063 //
00064 // Notes to maintainers:
00065 //   - Each macro here is a user-tweakable knob; do not grow the list
00066 //     lightly.
00067 //   - Use #if to key off these macros. Don't use #ifdef or "#if
00068 //     defined(...)", which will not work as these macros are ALWAYS
00069 //     defined.
00070 //
00071 //   GTEST_HAS_CLONE      - Define it to 1/0 to indicate that clone(2)
00072 //                         is/isn't available.
00073 //   GTEST_HAS_EXCEPTIONS - Define it to 1/0 to indicate that exceptions
00074 //                         are enabled.
00075 //   GTEST_HAS_GLOBAL_STRING - Define it to 1/0 to indicate that ::string
00076 //                             is/isn't available
00077 //   GTEST_HAS_GLOBAL_WSTRING - Define it to 1/0 to indicate that ::wstring
00078 //                             is/isn't available
00079 //   GTEST_HAS_POSIX_REGEX - Define it to 1/0 to indicate that POSIX regular
00080 //                           expressions are/aren't available.
00081 //   GTEST_HAS_PTHREAD      - Define it to 1/0 to indicate that <pthread.h>
00082 //                             is/isn't available.
00083 //   GTEST_HAS_RTTI         - Define it to 1/0 to indicate that RTTI is/isn't
00084 //                           enabled.
00085 //   GTEST_HAS_STD_WSTRING - Define it to 1/0 to indicate that
00086 //                           std::wstring does/doesn't work (Google Test can
00087 //                           be used where std::wstring is unavailable).
00088 //   GTEST_HAS_TRL_TUPLE    - Define it to 1/0 to indicate tr1::tuple
00089 //                           is/isn't available.
00090 //   GTEST_HAS_SEH          - Define it to 1/0 to indicate whether the
00091 //                           compiler supports Microsoft's "Structured
00092 //                           Exception Handling".
00093 //   GTEST_HAS_STREAM_REDIRECTION
00094 //           - Define it to 1/0 to indicate whether the
00095 //             platform supports I/O stream redirection using
00096 //             dup() and dup2().
00097 //   GTEST_USE_OWN_TRL_TUPLE - Define it to 1/0 to indicate whether Google
00098 //                           Test's own tr1 tuple implementation should be
00099 //                           used. Unused when the user sets
00100 //                           GTEST_HAS_TRL_TUPLE to 0.
00101 //   GTEST_LANG_CXX11       - Define it to 1/0 to indicate that Google Test
00102 //                           is building in C++11/C++98 mode.
00103 //   GTEST_LINKED_AS_SHARED_LIBRARY
00104 //           - Define to 1 when compiling tests that use
00105 //             Google Test as a shared library (known as
00106 //             DLL on Windows).
00107 //   GTEST_CREATE_SHARED_LIBRARY
00108 //           - Define to 1 when compiling Google Test itself
00109 //             as a shared library.
00110 //   GTEST_DEFAULT_DEATH_TEST_STYLE
00111 //           - The default value of --gtest_death_test_style.
00112 //             The legacy default has been "fast" in the open
00113 //             source version since 2008. The recommended value
00114 //             is "threadsafe", and can be set in
00115 //             custom/gtest-port.h.
00116
00117 // Platform-indicating macros
00118 // -----
00119 //
00120 // Macros indicating the platform on which Google Test is being used
00121 // (a macro is defined to 1 if compiled on the given platform;
00122 // otherwise UNDEFINED -- it's never defined to 0.). Google Test
00123 // defines these macros automatically. Code outside Google Test MUST
```

```

00124 // NOT define them.
00125 //
00126 //   GTEST_OS_AIX      - IBM AIX
00127 //   GTEST_OS_CYGWIN   - Cygwin
00128 //   GTEST_OS_FREEBSD  - FreeBSD
00129 //   GTEST_OS_FUCHSIA  - Fuchsia
00130 //   GTEST_OS_HPUX     - HP-UX
00131 //   GTEST_OS_LINUX    - Linux
00132 //   GTEST_OS_LINUX_ANDROID - Google Android
00133 //   GTEST_OS_MAC      - Mac OS X
00134 //   GTEST_OS_IOS      - iOS
00135 //   GTEST_OS_NACL     - Google Native Client (NaCl)
00136 //   GTEST_OS_NETBSD   - NetBSD
00137 //   GTEST_OS_OPENBSD  - OpenBSD
00138 //   GTEST_OS_QNX      - QNX
00139 //   GTEST_OS_SOLARIS  - Sun Solaris
00140 //   GTEST_OS_SYMBIAN  - Symbian
00141 //   GTEST_OS_WINDOWS   - Windows (Desktop, MinGW, or Mobile)
00142 //     GTEST_OS_WINDOWS_DESKTOP - Windows Desktop
00143 //     GTEST_OS_WINDOWS_MINGW  - MinGW
00144 //     GTEST_OS_WINDOWS_MOBILE - Windows Mobile
00145 //     GTEST_OS_WINDOWS_PHONE - Windows Phone
00146 //     GTEST_OS_WINDOWS_RT   - Windows Store App/WinRT
00147 //     GTEST_OS_ZOS        - z/OS
00148 //
00149 // Among the platforms, Cygwin, Linux, Max OS X, and Windows have the
00150 // most stable support. Since core members of the Google Test project
00151 // don't have access to other platforms, support for them may be less
00152 // stable. If you notice any problems on your platform, please notify
00153 // googletestframework@googlegroups.com (patches for fixing them are
00154 // even more welcome!).
00155 //
00156 // It is possible that none of the GTEST_OS_* macros are defined.
00157
00158 // Feature-indicating macros
00159 // -----
00160 //
00161 // Macros indicating which Google Test features are available (a macro
00162 // is defined to 1 if the corresponding feature is supported;
00163 // otherwise UNDEFINED -- it's never defined to 0.). Google Test
00164 // defines these macros automatically. Code outside Google Test MUST
00165 // NOT define them.
00166 //
00167 // These macros are public so that portable tests can be written.
00168 // Such tests typically surround code using a feature with an #if
00169 // which controls that code. For example:
00170 //
00171 // #if GTEST_HAS_DEATH_TEST
00172 //   EXPECT_DEATH(DoSomethingDeadly());
00173 // #endif
00174 //
00175 //   GTEST_HAS_COMBINE      - the Combine() function (for value-parameterized
00176 //                           tests)
00177 //   GTEST_HAS_DEATH_TEST   - death tests
00178 //   GTEST_HAS_TYPED_TEST   - typed tests
00179 //   GTEST_HAS_TYPED_TEST_P - type-parameterized tests
00180 //   GTEST_IS_THREADSAFE   - Google Test is thread-safe.
00181 // GOOGLETEST_CM0007 DO NOT DELETE
00182 //   GTEST_USES_POSIX_REGEX - enhanced POSIX regex is used. Do not confuse with
00183 //                           GTEST_HAS_POSIX_REGEX (see above) which users can
00184 //                           define themselves.
00185 //   GTEST_USES_SIMPLE_REGEX - our own simple regex is used;
00186 //                           the above RE\b(s) are mutually exclusive.
00187 //   GTEST_CAN_COMPARE_NULL - accepts untyped NULL in EXPECT_EQ().
00188
00189 // Misc public macros
00190 // -----
00191 //
00192 //   GTEST_FLAG(flag_name) - references the variable corresponding to
00193 //                           the given Google Test flag.
00194
00195 // Internal utilities
00196 // -----
00197 //
00198 // The following macros and utilities are for Google Test's INTERNAL
00199 // use only. Code outside Google Test MUST NOT USE THEM DIRECTLY.
00200 //
00201 // Macros for basic C++ coding:
00202 //   GTEST_AMBIGUOUS_ELSE_BLOCKER_ - for disabling a gcc warning.
00203 //   GTEST_ATTRIBUTE_UNUSED_ - declares that a class' instances or a
00204 //                           variable don't have to be used.
00205 //   GTEST_DISALLOW_ASSIGN_ - disables operator=.
00206 //   GTEST_DISALLOW_COPY_AND_ASSIGN_ - disables copy ctor and operator=.
00207 //   GTEST_MUST_USE_RESULT_ - declares that a function's result must be used.
00208 //   GTEST_INTENTIONAL_CONST_COND_PUSH_ - start code section where MSVC C4127 is
00209 //                                         suppressed (constant conditional).
00210 //   GTEST_INTENTIONAL_CONST_COND_POP_ - finish code section where MSVC C4127

```

```
00211 // is suppressed.
00212 //
00213 // C++11 feature wrappers:
00214 //
00215 // testing::internal::forward - portability wrapper for std::forward.
00216 // testing::internal::move - portability wrapper for std::move.
00217 //
00218 // Synchronization:
00219 // Mutex, MutexLock, ThreadLocal, GetThreadCount()
00220 // - synchronization primitives.
00221 //
00222 // Template meta programming:
00223 // is_pointer - as in TR1; needed on Symbian and IBM XL C/C++ only.
00224 // IteratorTraits - partial implementation of std::iterator_traits, which
00225 // is not available in libCstd when compiled with Sun C++.
00226 //
00227 // Smart pointers:
00228 // scoped_ptr - as in TR2.
00229 //
00230 // Regular expressions:
00231 // RE - a simple regular expression class using the POSIX
00232 // Extended Regular Expression syntax on UNIX-like platforms
00233 // GOOGLETEST_CM0008 DO NOT DELETE
00234 // or a reduced regular exception syntax on other
00235 // platforms, including Windows.
00236 // Logging:
00237 // GTEST_LOG_(*) - logs messages at the specified severity level.
00238 // LogToStderr() - directs all log messages to stderr.
00239 // FlushInfoLog() - flushes informational log messages.
00240 //
00241 // Stdout and stderr capturing:
00242 // CaptureStdout() - starts capturing stdout.
00243 // GetCapturedStdout() - stops capturing stdout and returns the captured
00244 // string.
00245 // CaptureStderr() - starts capturing stderr.
00246 // GetCapturedStderr() - stops capturing stderr and returns the captured
00247 // string.
00248 //
00249 // Integer types:
00250 // TypeWithSize - maps an integer to a int type.
00251 // Int32, UInt32, Int64, UInt64, TimeInMillis
00252 // - integers of known sizes.
00253 // BiggestInt - the biggest signed integer type.
00254 //
00255 // Command-line utilities:
00256 // GTEST_DECLARE_*(*) - declares a flag.
00257 // GTEST_DEFINE_*(*) - defines a flag.
00258 // GetInjectableArgvs() - returns the command line as a vector of strings.
00259 //
00260 // Environment variable utilities:
00261 // GetEnv() - gets the value of an environment variable.
00262 // BoolFromGTestEnv() - parses a bool environment variable.
00263 // Int32FromGTestEnv() - parses an Int32 environment variable.
00264 // StringFromGTestEnv() - parses a string environment variable.
00265
00266 #include <ctype.h> // for isspace, etc
00267 #include <stddef.h> // for ptrdiff_t
00268 #include <stdlib.h>
00269 #include <stdio.h>
00270 #include <string.h>
00271 #ifndef __WIN32_WCE
00272 # include <sys/types.h>
00273 # include <sys/stat.h>
00274 #endif // __WIN32_WCE
00275
00276 #if defined __APPLE__
00277 # include <AvailabilityMacros.h>
00278 # include <TargetConditionals.h>
00279 #endif
00280
00281 // Brings in the definition of HAS_GLOBAL_STRING. This must be done
00282 // BEFORE we test HAS_GLOBAL_STRING.
00283 #include <string> // NOLINT
00284 #include <algorithm> // NOLINT
00285 #include <iostream> // NOLINT
00286 #include <sstream> // NOLINT
00287 #include <utility>
00288 #include <vector> // NOLINT
00289
00290 #include "gtest/internal/gtest-port-arch.h"
00291 #include "gtest/internal/custom/gtest-port.h"
00292
00293 #if !defined(GTEST_DEV_EMAIL_)
00294 # define GTEST_DEV_EMAIL_ "googletestframework@googlegroups.com"
00295 # define GTEST_FLAG_PREFIX_ "gtest_"
00296 # define GTEST_FLAG_PREFIX_DASH_ "gtest-"
00297 # define GTEST_FLAG_PREFIX_UPPER_ "GTEST_"
```

```

00298 # define GTEST_NAME_ "Google Test"
00299 # define GTEST_PROJECT_URL_ "https://github.com/google/googletest/"
00300 #endif // !defined(GTEST_DEV_EMAIL_)
00301
00302 #if !defined(GTEST_INIT_GOOGLE_TEST_NAME_)
00303 # define GTEST_INIT_GOOGLE_TEST_NAME_ "testing::InitGoogleTest"
00304 #endif // !defined(GTEST_INIT_GOOGLE_TEST_NAME_)
00305
00306 // Determines the version of gcc that is used to compile this.
00307 #ifdef __GNUC__
00308 // 40302 means version 4.3.2.
00309 # define GTEST_GCC_VER_ \
00310     (__GNUC__*10000 + __GNUC_MINOR__*100 + __GNUC_PATCHLEVEL__)
00311 #endif // __GNUC__
00312
00313 // Macros for disabling Microsoft Visual C++ warnings.
00314 //
00315 //    GTEST_DISABLE_MSC_WARNINGS_PUSH_(4800 4385)
00316 //    /* code that triggers warnings C4800 and C4385 */
00317 //    GTEST_DISABLE_MSC_WARNINGS_POP_()
00318 #if _MSC_VER >= 1400
00319 # define GTEST_DISABLE_MSC_WARNINGS_PUSH_(warnings) \
00320     __pragma(warning(push)) \
00321     __pragma(warning(disable: warnings)) \
00322 # define GTEST_DISABLE_MSC_WARNINGS_POP_() \
00323     __pragma(warning(pop))
00324 #else
00325 // Older versions of MSVC don't have __pragma.
00326 # define GTEST_DISABLE_MSC_WARNINGS_PUSH_(warnings)
00327 # define GTEST_DISABLE_MSC_WARNINGS_POP_()
00328 #endif
00329
00330 // Clang on Windows does not understand MSVC's pragma warning.
00331 // We need clang-specific way to disable function deprecation warning.
00332 #ifdef __clang__
00333 # define GTEST_DISABLE_MSC_DEPRECATED_PUSH_()
00334     __Pragma("clang diagnostic push") \
00335     __Pragma("clang diagnostic ignored \"-Wdeprecated-declarations\"") \
00336     __Pragma("clang diagnostic ignored \"-Wdeprecated-implementations\"")
00337 #define GTEST_DISABLE_MSC_DEPRECATED_POP_()
00338     __Pragma("clang diagnostic pop")
00339 #else
00340 # define GTEST_DISABLE_MSC_DEPRECATED_PUSH_()
00341     GTEST_DISABLE_MSC_WARNINGS_PUSH_(4996)
00342 # define GTEST_DISABLE_MSC_DEPRECATED_POP_()
00343     GTEST_DISABLE_MSC_WARNINGS_POP_()
00344 #endif
00345
00346 #ifndef GTEST_LANG_CXX11
00347 // gcc and clang define __GXX_EXPERIMENTAL_CXX0X__ when
00348 // -std={c,gnu}++{0x,11} is passed. The C++11 standard specifies a
00349 // value for __cplusplus, and recent versions of clang, gcc, and
00350 // probably other compilers set that too in C++11 mode.
00351 # if __GXX_EXPERIMENTAL_CXX0X__ || __cplusplus >= 201103L || _MSC_VER >= 1900
00352 // Compiling in at least C++11 mode.
00353 # define GTEST_LANG_CXX11 1
00354 # else
00355 # define GTEST_LANG_CXX11 0
00356 # endif
00357 #endif
00358
00359 // Distinct from C++11 language support, some environments don't provide
00360 // proper C++11 library support. Notably, it's possible to build in
00361 // C++11 mode when targeting Mac OS X 10.6, which has an old libstdc++
00362 // with no C++11 support.
00363 //
00364 // libstdc++ has sufficient C++11 support as of GCC 4.6.0, __GLIBCXX__
00365 // 20110325, but maintenance releases in the 4.4 and 4.5 series followed
00366 // this date, so check for those versions by their date stamps.
00367 // https://gcc.gnu.org/onlinedocs/libstdc++/manual/abi.html#abi.versioning
00368 #if GTEST_LANG_CXX11 && \
00369     (!defined(__GLIBCXX__) || ( \
00370         __GLIBCXX__ >= 20110325ul && /* GCC >= 4.6.0 */ \
00371         /* Denylist of patch releases of older branches: */ \
00372         __GLIBCXX__ != 20110416ul && /* GCC 4.4.6 */ \
00373         __GLIBCXX__ != 20120313ul && /* GCC 4.4.7 */ \
00374         __GLIBCXX__ != 20110428ul && /* GCC 4.5.3 */ \
00375         __GLIBCXX__ != 20120702ul)) /* GCC 4.5.4 */
00376 # define GTEST_STDLIB_CXX11 1
00377 #endif
00378
00379 // Only use C++11 library features if the library provides them.
00380 #if GTEST_STDLIB_CXX11
00381 # define GTEST_HAS_STD_BEGIN_AND_END_ 1
00382 # define GTEST_HAS_STD_FORWARD_LIST_ 1
00383 # if !defined(_MSC_VER) || (_MSC_FULL_VER >= 190023824)
00384 // works only with VS2015U2 and better

```

```
00385 # define GTEST_HAS_STD_FUNCTION_ 1
00386 # endif
00387 # define GTEST_HAS_STD_INITIALIZER_LIST_ 1
00388 # define GTEST_HAS_STD_MOVE_ 1
00389 # define GTEST_HAS_STD_UNIQUE_PTR_ 1
00390 # define GTEST_HAS_STD_SHARED_PTR_ 1
00391 # define GTEST_HAS_UNORDERED_MAP_ 1
00392 # define GTEST_HAS_UNORDERED_SET_ 1
00393 #endif
00394
00395 // C++11 specifies that <tuple> provides std::tuple.
00396 // Some platforms still might not have it, however.
00397 #if GTEST_LANG_CXX11
00398 # define GTEST_HAS_STD_TUPLE_ 1
00399 # if defined(__clang__)
00400 // Inspired by
00401 // https://clang.llvm.org/docs/LanguageExtensions.html#include-file-checking-macros
00402 # if defined(_has_include) && !_has_include(<tuple>)
00403 # undef GTEST_HAS_STD_TUPLE_
00404 # endif
00405 # elif defined(_MSC_VER)
00406 // Inspired by boost/config/stdlib/dinkumware.hpp
00407 # if defined(_CPPLIB_VER) && _CPPLIB_VER < 520
00408 # undef GTEST_HAS_STD_TUPLE_
00409 # endif
00410 # elif defined(__GLIBCXX__)
00411 // Inspired by boost/config/stdlib/libstdcpp3.hpp,
00412 // http://gcc.gnu.org/gcc-4.2/changes.html and
00413 //
00414 // https://web.archive.org/web/20140227044429/gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt01ch01.html#manual.intro.status
00415 # if __GNUC__ < 4 || (__GNUC__ == 4 && __GNUC_MINOR__ < 2)
00416 # undef GTEST_HAS_STD_TUPLE_
00417 # endif
00418 #endif
00419
00420 // Brings in definitions for functions used in the testing::internal::posix
00421 // namespace (read, write, close, chdir, isatty, stat). We do not currently
00422 // use them on Windows Mobile.
00423 #if GTEST_OS_WINDOWS
00424 # if !GTEST_OS_WINDOWS_MOBILE
00425 # include <direct.h>
00426 # include <iostream>
00427 #endif
00428 // In order to avoid having to include <windows.h>, use forward declaration
00429 #if GTEST_OS_WINDOWS_MINGW && !defined(__MINGW64_VERSION_MAJOR)
00430 // MinGW defined _CRITICAL_SECTION and _RTL_CRITICAL_SECTION as two
00431 // separate (equivalent) structs, instead of using typedef
00432 typedef struct _CRITICAL_SECTION GTEST_CRITICAL_SECTION;
00433 #else
00434 // Assume CRITICAL_SECTION is a typedef of _RTL_CRITICAL_SECTION.
00435 // This assumption is verified by
00436 // WindowsTypesTest.CRITICAL_SECTIONIS_RTL_CRITICAL_SECTION.
00437 typedef struct _RTL_CRITICAL_SECTION GTEST_CRITICAL_SECTION;
00438 #endif
00439 #else
00440 // This assumes that non-Windows OSes provide unistd.h. For OSes where this
00441 // is not the case, we need to include headers that provide the functions
00442 // mentioned above.
00443 # include <unistd.h>
00444 # include <strings.h>
00445 #endif // GTEST_OS_WINDOWS
00446
00447 #if GTEST_OS_LINUX_ANDROID
00448 // Used to define __ANDROID_API__ matching the target NDK API level.
00449 # include <android/api-level.h> // NOLINT
00450 #endif
00451
00452 // Defines this to true iff Google Test can use POSIX regular expressions.
00453 #ifndef GTEST_HAS_POSIX_RE
00454 # if GTEST_OS_LINUX_ANDROID
00455 // On Android, <regex.h> is only available starting with Gingerbread.
00456 # define GTEST_HAS_POSIX_RE (__ANDROID_API__ >= 9)
00457 # else
00458 # define GTEST_HAS_POSIX_RE (!GTEST_OS_WINDOWS)
00459 # endif
00460 #endif
00461
00462 #if GTEST_USES_PCRE
00463 // The appropriate headers have already been included.
00464
00465 #elif GTEST_HAS_POSIX_RE
00466
00467 // On some platforms, <regex.h> needs someone to define size_t, and
00468 // won't compile otherwise. We can #include it here as we already
00469 // included <stdlib.h>, which is guaranteed to define size_t through
00470 // <stddef.h>.
```

```
00471 # include <regex.h> // NOLINT
00472
00473 # define GTEST_USES_POSIX_RE 1
00474
00475 #elif GTEST_OS_WINDOWS
00476
00477 // <regex.h> is not available on Windows. Use our own simple regex
00478 // implementation instead.
00479 # define GTEST_USES_SIMPLE_RE 1
00480
00481 #else
00482
00483 // <regex.h> may not be available on this platform. Use our own
00484 // simple regex implementation instead.
00485 # define GTEST_USES_SIMPLE_RE 1
00486
00487 #endif // GTEST_USES_PCRE
00488
00489 #ifndef GTEST_HAS_EXCEPTIONS
00490 // The user didn't tell us whether exceptions are enabled, so we need
00491 // to figure it out.
00492 # if defined(_MSC_VER) && defined(_CPPUNWIND)
00493 // MSVC defines _CPPUNWIND to 1 iff exceptions are enabled.
00494 # define GTEST_HAS_EXCEPTIONS 1
00495 # elif defined(__BORLANDC__)
00496 // C++Builder's implementation of the STL uses the __HAS_EXCEPTIONS
00497 // macro to enable exceptions, so we'll do the same.
00498 // Assumes that exceptions are enabled by default.
00499 # ifndef __HAS_EXCEPTIONS
00500 # define __HAS_EXCEPTIONS 1
00501 # endif // __HAS_EXCEPTIONS
00502 # define GTEST_HAS_EXCEPTIONS __HAS_EXCEPTIONS
00503 # elif defined(__clang__)
00504 // clang defines __EXCEPTIONS iff exceptions are enabled before clang 220714,
00505 // but iff cleanups are enabled after that. In Obj-C++ files, there can be
00506 // cleanups for ObjC exceptions which also need cleanups, even if C++ exceptions
00507 // are disabled. clang has __has_feature(cxx_exceptions) which checks for C++
00508 // exceptions starting at clang r206352, but which checked for cleanups prior to
00509 // that. To reliably check for C++ exception availability with clang, check for
00510 // __EXCEPTIONS && __has_feature(cxx_exceptions).
00511 # define GTEST_HAS_EXCEPTIONS (__EXCEPTIONS && __has_feature(cxx_exceptions))
00512 # elif defined(__GNUC__) && __EXCEPTIONS
00513 // gcc defines __EXCEPTIONS to 1 iff exceptions are enabled.
00514 # define GTEST_HAS_EXCEPTIONS 1
00515 # elif defined(__SUNPRO_CC)
00516 // Sun Pro CC supports exceptions. However, there is no compile-time way of
00517 // detecting whether they are enabled or not. Therefore, we assume that
00518 // they are enabled unless the user tells us otherwise.
00519 # define GTEST_HAS_EXCEPTIONS 1
00520 # elif defined(__IBMCPP__) && __EXCEPTIONS
00521 // xlC defines __EXCEPTIONS to 1 iff exceptions are enabled.
00522 # define GTEST_HAS_EXCEPTIONS 1
00523 # elif defined(__HP_aCC)
00524 // Exception handling is in effect by default in HP aCC compiler. It has to
00525 // be turned off by +noeh compiler option if desired.
00526 # define GTEST_HAS_EXCEPTIONS 1
00527 # else
00528 // For other compilers, we assume exceptions are disabled to be
00529 // conservative.
00530 # define GTEST_HAS_EXCEPTIONS 0
00531 # endif // defined(_MSC_VER) || defined(__BORLANDC__)
00532 #endif // GTEST_HAS_EXCEPTIONS
00533
00534 #if !defined(GTEST_HAS_STD_STRING)
00535 // Even though we don't use this macro any longer, we keep it in case
00536 // some clients still depend on it.
00537 # define GTEST_HAS_STD_STRING 1
00538 #elif !GTEST_HAS_STD_STRING
00539 // The user told us that ::std::string isn't available.
00540 # error "::std::string isn't available."
00541 #endif // !defined(GTEST_HAS_STD_STRING)
00542
00543 #ifndef GTEST_HAS_GLOBAL_STRING
00544 # define GTEST_HAS_GLOBAL_STRING 0
00545 #endif // GTEST_HAS_GLOBAL_STRING
00546
00547 #ifndef GTEST_HAS_STD_WSTRING
00548 // The user didn't tell us whether ::std::wstring is available, so we need
00549 // to figure it out.
00550 // FIXME: uses autoconf to detect whether ::std::wstring
00551 // is available.
00552
00553 // Cygwin 1.7 and below doesn't support ::std::wstring.
00554 // Solaris' libc++ doesn't support it either. Android has
00555 // no support for it at least as recent as Froyo (2.2).
00556 # define GTEST_HAS_STD_WSTRING \
00557     (! (GTEST_OS_LINUX_ANDROID || GTEST_OS_CYGWIN || GTEST_OS_SOLARIS))
```

```
00558
00559 #endif // GTEST_HAS_STD_WSTRING
00560
00561 #ifndef GTEST_HAS_GLOBAL_WSTRING
00562 // The user didn't tell us whether ::wstring is available, so we need
00563 // to figure it out.
00564 # define GTEST_HAS_GLOBAL_WSTRING \
00565     (GTEST_HAS_STD_WSTRING && GTEST_HAS_GLOBAL_STRING)
00566 #endif // GTEST_HAS_GLOBAL_WSTRING
00567
00568 // Determines whether RTTI is available.
00569 #ifndef GTEST_HAS_RTTI
00570 // The user didn't tell us whether RTTI is enabled, so we need to
00571 // figure it out.
00572
00573 # ifdef _MSC_VER
00574
00575 # ifdef _CPPRTTI // MSVC defines this macro iff RTTI is enabled.
00576 # define GTEST_HAS_RTTI 1
00577 # else
00578 # define GTEST_HAS_RTTI 0
00579 # endif
00580
00581 // Starting with version 4.3.2, gcc defines __GXX_RTTI iff RTTI is enabled.
00582 # elif defined(__GNUC__) && (GTEST_GCC_VER_ >= 40302)
00583
00584 # ifdef __GXX_RTTI
00585 // When building against STLport with the Android NDK and with
00586 // -frtti -fno-exceptions, the build fails at link time with undefined
00587 // references to __cxa_bad_typeid. Note sure if STL or toolchain bug,
00588 // so disable RTTI when detected.
00589 # if GTEST_OS_LINUX_ANDROID && defined(_STLPORT_MAJOR) && \
00590     !defined(__EXCEPTIONS)
00591 # define GTEST_HAS_RTTI 0
00592 # else
00593 # define GTEST_HAS_RTTI 1
00594 # endif // GTEST_OS_LINUX_ANDROID && __STLPORT_MAJOR && !__EXCEPTIONS
00595 # else
00596 # define GTEST_HAS_RTTI 0
00597 # endif // __GXX_RTTI
00598
00599 // Clang defines __GXX_RTTI starting with version 3.0, but its manual recommends
00600 // using has_feature instead. has_feature(cxx_rtti) is supported since 2.7, the
00601 // first version with C++ support.
00602 # elif defined(__clang__)
00603
00604 # define GTEST_HAS_RTTI __has_feature(cxx_rtti)
00605
00606 // Starting with version 9.0 IBM Visual Age defines __RTTI_ALL__ to 1 if
00607 // both the typeid and dynamic_cast features are present.
00608 # elif defined(__IBMCPP__) && (__IBMCPP__ >= 900)
00609
00610 # ifdef __RTTI_ALL__
00611 # define GTEST_HAS_RTTI 1
00612 # else
00613 # define GTEST_HAS_RTTI 0
00614 # endif
00615
00616 # else
00617
00618 // For all other compilers, we assume RTTI is enabled.
00619 # define GTEST_HAS_RTTI 1
00620
00621 # endif // _MSC_VER
00622
00623 #endif // GTEST_HAS_RTTI
00624
00625 // It's this header's responsibility to #include <typeinfo> when RTTI
00626 // is enabled.
00627 #if GTEST_HAS_RTTI
00628 # include <typeinfo>
00629 #endif
00630
00631 // Determines whether Google Test can use the pthreads library.
00632 #ifndef GTEST_HAS_PTHREAD
00633 // The user didn't tell us explicitly, so we make reasonable assumptions about
00634 // which platforms have pthreads support.
00635 //
00636 // To disable threading support in Google Test, add -DGTEST_HAS_PTHREAD=0
00637 // to your compiler flags.
00638 #define GTEST_HAS_PTHREAD \
00639     (GTEST_OS_LINUX || GTEST_OS_MAC || GTEST_OS_HPUX || GTEST_OS_QNX || \
00640     GTEST_OS_FREEBSD || GTEST_OS_NACL || GTEST_OS_NETBSD || GTEST_OS_FUCHSIA)
00641 #endif // GTEST_HAS_PTHREAD
00642
00643 #if GTEST_HAS_PTHREAD
00644 // gtest-port.h guarantees to #include <pthread.h> when GTEST_HAS_PTHREAD is
```

```
00645 // true.
00646 # include <pthread.h> // NOLINT
00647
00648 // For timespec and nanosleep, used below.
00649 # include <time.h> // NOLINT
00650 #endif
00651
00652 // Determines if hash_map/hash_set are available.
00653 // Only used for testing against those containers.
00654 #if !defined(GTEST_HAS_HASH_MAP_)
00655 # if defined(_MSC_VER) && (_MSC_VER < 1900)
00656 # define GTEST_HAS_HASH_MAP_ 1 // Indicates that hash_map is available.
00657 # define GTEST_HAS_HASH_SET_ 1 // Indicates that hash_set is available.
00658 # endif // _MSC_VER
00659 #endif // !defined(GTEST_HAS_HASH_MAP_)
00660
00661 // Determines whether Google Test can use tr1/tuple. You can define
00662 // this macro to 0 to prevent Google Test from using tuple (any
00663 // feature depending on tuple will be disabled in this mode).
00664 #ifndef GTEST_HAS_TR1_TUPLE
00665 # if GTEST_OS_LINUX_ANDROID && defined(_STLPORT_MAJOR)
00666 // STLport, provided with the Android NDK, has neither <tr1/tuple> or <tuple>.
00667 # define GTEST_HAS_TR1_TUPLE 0
00668 # elif defined(_MSC_VER) && (_MSC_VER >= 1910)
00669 // Prevent `warning C4996: 'std::tr1': warning STL4002:
00670 // The non-Standard std::tr1 namespace and TR1-only machinery
00671 // are deprecated and will be REMOVED.`
00672 # define GTEST_HAS_TR1_TUPLE 0
00673 # elif GTEST_LANG_CXX11 && defined(_LIBCPP_VERSION)
00674 // libcpp doesn't support TR1.
00675 # define GTEST_HAS_TR1_TUPLE 0
00676 # else
00677 // The user didn't tell us not to do it, so we assume it's OK.
00678 # define GTEST_HAS_TR1_TUPLE 1
00679 # endif
00680 #endif // GTEST_HAS_TR1_TUPLE
00681
00682 // Determines whether Google Test's own tr1 tuple implementation
00683 // should be used.
00684 #ifndef GTEST_USE_OWN_TR1_TUPLE
00685 // We use our own tuple implementation on Symbian.
00686 # if GTEST_OS_SYMBIAN
00687 # define GTEST_USE_OWN_TR1_TUPLE 1
00688 # else
00689 // The user didn't tell us, so we need to figure it out.
00690
00691 // We use our own TR1 tuple if we aren't sure the user has an
00692 // implementation of it already. At this time, libstdc++ 4.0.0+ and
00693 // MSVC 2010 are the only mainstream standard libraries that come
00694 // with a TR1 tuple implementation. NVIDIA's CUDA NVCC compiler
00695 // pretends to be GCC by defining __GNUC__ and friends, but cannot
00696 // compile GCC's tuple implementation. MSVC 2008 (9.0) provides TR1
00697 // tuple in a 323 MB Feature Pack download, which we cannot assume the
00698 // user has. QNX's QCC compiler is a modified GCC but it doesn't
00699 // support TR1 tuple. libcpp only provides std::tuple, in C++11 mode,
00700 // and it can be used with some compilers that define __GNUC__.
00701 # if (defined(__GNUC__) && !defined(__CUDACC__)) && (GTEST_GCC_VER_ >= 40000) \
00702     && !GTEST_OS_QNX && !defined(_LIBCPP_VERSION)) \
00703     || (_MSC_VER >= 1600 && _MSC_VER < 1900)
00704 # define GTEST_ENV_HAS_TR1_TUPLE_ 1
00705 # endif
00706
00707 // C++11 specifies that <tuple> provides std::tuple. Use that if gtest is used
00708 // in C++11 mode and libstdc++ isn't very old (binaries targeting OS X 10.6
00709 // can build with clang but need to use gcc4.2's libstdc++).
00710 # if GTEST_LANG_CXX11 && (!defined(__GLIBCXX__) || __GLIBCXX__ > 20110325)
00711 # define GTEST_ENV_HAS_STD_TUPLE_ 1
00712 # endif
00713
00714 # if GTEST_ENV_HAS_TR1_TUPLE_ || GTEST_ENV_HAS_STD_TUPLE_
00715 # define GTEST_USE_OWN_TR1_TUPLE 0
00716 # else
00717 # define GTEST_USE_OWN_TR1_TUPLE 1
00718 # endif
00719 # endif // GTEST_OS_SYMBIAN
00720 #endif // GTEST_USE_OWN_TR1_TUPLE
00721
00722 // To avoid conditional compilation we make it gtest-port.h's responsibility
00723 // to #include the header implementing tuple.
00724 #if GTEST_HAS_STD_TUPLE_
00725 # include <tuple> // IWYU pragma: export
00726 # define GTEST_TUPLE_NAMESPACE_ ::std
00727 #endif // GTEST_HAS_STD_TUPLE_
00728
00729 // We include tr1::tuple even if std::tuple is available to define printers for
00730 // them.
00731 #if GTEST_HAS_TR1_TUPLE
```

```
00732 # ifndef GTEST_TUPLE_NAMESPACE_
00733 #  define GTEST_TUPLE_NAMESPACE_ ::std::tr1
00734 # endif // GTEST_TUPLE_NAMESPACE_
00735
00736 # if GTEST_USE_OWN_TR1_TUPLE
00737 #  include "gtest/internal/gtest-tuple.h" // IWYU pragma: export // NOLINT
00738 # elif GTEST_OS_SYMBIAN
00739
00740 // On Symbian, BOOST_HAS_TR1_TUPLE causes Boost's TR1 tuple library to
00741 // use STLport's tuple implementation, which unfortunately doesn't
00742 // work as the copy of STLport distributed with Symbian is incomplete.
00743 // By making sure BOOST_HAS_TR1_TUPLE is undefined, we force Boost to
00744 // use its own tuple implementation.
00745 # ifdef BOOST_HAS_TR1_TUPLE
00746 #  undef BOOST_HAS_TR1_TUPLE
00747 # endif // BOOST_HAS_TR1_TUPLE
00748
00749 // This prevents <boost/tr1/detail/config.hpp>, which defines
00750 // BOOST_HAS_TR1_TUPLE, from being #included by Boost's <tuple>.
00751 # define BOOST_TR1_DETAIL_CONFIG_HPP_INCLUDED
00752 # include <tuple> // IWYU pragma: export // NOLINT
00753
00754 # elif defined(__GNUC__) && (GTEST_GCC_VER_ >= 40000)
00755 // GCC 4.0+ implements tr1/tuple in the <tr1/tuple> header. This does
00756 // not conform to the TR1 spec, which requires the header to be <tuple>.
00757
00758 # if !GTEST_HAS_RTTI && GTEST_GCC_VER_ < 40302
00759 // Until version 4.3.2, gcc has a bug that causes <tr1/functional>,
00760 // which is #included by <tr1/tuple>, to not compile when RTTI is
00761 // disabled. _TR1_FUNCTIONAL is the header guard for
00762 // <tr1/functional>. Hence the following #define is used to prevent
00763 // <tr1/functional> from being included.
00764 #  define _TR1_FUNCTIONAL 1
00765 #  include <tr1/tuple>
00766 #  undef _TR1_FUNCTIONAL // Allows the user to #include
00767 // <tr1/functional> if they choose to.
00768 # else
00769 #  include <tr1/tuple> // NOLINT
00770 # endif // !GTEST_HAS_RTTI && GTEST_GCC_VER_ < 40302
00771
00772 // VS 2010 now has tr1 support.
00773 # elif __MSC_VER >= 1600
00774 #  include <tuple> // IWYU pragma: export // NOLINT
00775
00776 # else // GTEST_USE_OWN_TR1_TUPLE
00777 #  include <tr1/tuple> // IWYU pragma: export // NOLINT
00778 # endif // GTEST_USE_OWN_TR1_TUPLE
00779
00780 #endif // GTEST_HAS_TR1_TUPLE
00781
00782 // Determines whether clone(2) is supported.
00783 // Usually it will only be available on Linux, excluding
00784 // Linux on the Itanium architecture.
00785 // Also see http://linux.die.net/man/2/clone.
00786 #ifndef GTEST_HAS_CLONE
00787 // The user didn't tell us, so we need to figure it out.
00788
00789 # if GTEST_OS_LINUX && !defined(__ia64__)
00790 #  if GTEST_OS_LINUX_ANDROID
00791 // On Android, clone() became available at different API levels for each 32-bit
00792 // architecture.
00793 #   if defined(__LP64__) || \
00794     (defined(__arm__) && __ANDROID_API__ >= 9) || \
00795     (defined(__mips__) && __ANDROID_API__ >= 12) || \
00796     (defined(__i386__) && __ANDROID_API__ >= 17)
00797 #     define GTEST_HAS_CLONE 1
00798 #   else
00799 #     define GTEST_HAS_CLONE 0
00800 #   endif
00801 # else
00802 #   define GTEST_HAS_CLONE 1
00803 # endif
00804 # else
00805 #   define GTEST_HAS_CLONE 0
00806 # endif // GTEST_OS_LINUX && !defined(__ia64__)
00807
00808 #endif // GTEST_HAS_CLONE
00809
00810 // Determines whether to support stream redirection. This is used to test
00811 // output correctness and to implement death tests.
00812 #ifndef GTEST_HAS_STREAM_REDIRECTION
00813 // By default, we assume that stream redirection is supported on all
00814 // platforms except known mobile ones.
00815 # if GTEST_OS_WINDOWS_MOBILE || GTEST_OS_SYMBIAN || \
00816     GTEST_OS_WINDOWS_PHONE || GTEST_OS_WINDOWS_RT
00817 #  define GTEST_HAS_STREAM_REDIRECTION 0
00818 # else
```

```

00819 # define GTEST_HAS_STREAM_REDIRECTION 1
00820 # endif // !GTEST_OS_WINDOWS_MOBILE && !GTEST_OS_SYMBIAN
00821 #endif // GTEST_HAS_STREAM_REDIRECTION
00822
00823 // Determines whether to support death tests.
00824 // Google Test does not support death tests for VC 7.1 and earlier as
00825 // abort() in a VC 7.1 application compiled as GUI in debug config
00826 // pops up a dialog window that cannot be suppressed programmatically.
00827 #if (GTEST_OS_LINUX || GTEST_OS_CYGIN || GTEST_OS_SOLARIS || \
00828     (GTEST_OS_MAC && !GTEST_OS_IOS) || \
00829     (GTEST_OS_WINDOWS_DESKTOP && _MSC_VER >= 1400) || \
00830     GTEST_OS_WINDOWS_MINGW || GTEST_OS_AIX || GTEST_OS_HPUX || \
00831     GTEST_OS_OPENBSD || GTEST_OS_QNX || GTEST_OS_FREEBSD || \
00832     GTEST_OS_NETBSD || GTEST_OS_FUCHSIA)
00833 # define GTEST_HAS_DEATH_TEST 1
00834 #endif
00835
00836 // Determines whether to support type-driven tests.
00837
00838 // Typed tests need <typeinfo> and variadic macros, which GCC, VC++ 8.0,
00839 // Sun Pro CC, IBM Visual Age, and HP aCC support.
00840 #if defined(__GNUC__) || (_MSC_VER >= 1400) || defined(__SUNPRO_CC) || \
00841     defined(__IBMCPP__) || defined(__HP_aCC)
00842 # define GTEST_HAS_TYPED_TEST 1
00843 # define GTEST_HAS_TYPED_TEST_P 1
00844 #endif
00845
00846 // Determines whether to support Combine(). This only makes sense when
00847 // value-parameterized tests are enabled. The implementation doesn't
00848 // work on Sun Studio since it doesn't understand templated conversion
00849 // operators.
00850 #if (GTEST_HAS_TR1_TUPLE || GTEST_HAS_STD_TUPLE_) && !defined(__SUNPRO_CC)
00851 # define GTEST_HAS_COMBINE 1
00852 #endif
00853
00854 // Determines whether the system compiler uses UTF-16 for encoding wide strings.
00855 #define GTEST_WIDE_STRING_USES_UTF16_ \
00856     (GTEST_OS_WINDOWS || GTEST_OS_CYGIN || GTEST_OS_SYMBIAN || GTEST_OS_AIX)
00857
00858 // Determines whether test results can be streamed to a socket.
00859 #if GTEST_OS_LINUX
00860 # define GTEST_CAN_STREAM_RESULTS_ 1
00861 #endif
00862
00863 // Defines some utility macros.
00864
00865 // The GNU compiler emits a warning if nested "if" statements are followed by
00866 // an "else" statement and braces are not used to explicitly disambiguate the
00867 // "else" binding. This leads to problems with code like:
00868 //
00869 // if (gate)
00870 //     ASSERT_*(condition) « "Some message";
00871 //
00872 // The "switch (0) case 0:" idiom is used to suppress this.
00873 #ifdef __INTEL_COMPILER
00874 # define GTEST_AMBIGUOUS_ELSE_BLOCKER_
00875 #else
00876 # define GTEST_AMBIGUOUS_ELSE_BLOCKER_ switch (0) case 0: default: // NOLINT
00877 #endif
00878
00879 // Use this annotation at the end of a struct/class definition to
00880 // prevent the compiler from optimizing away instances that are never
00881 // used. This is useful when all interesting logic happens inside the
00882 // c'tor and / or d'tor. Example:
00883 //
00884 // struct Foo {
00885 //     Foo() { ... }
00886 // } GTEST_ATTRIBUTE_UNUSED_;
00887 //
00888 // Also use it after a variable or parameter declaration to tell the
00889 // compiler the variable/parameter does not have to be used.
00890 #if defined(__GNUC__) && !defined(COMPILER_ICC)
00891 # define GTEST_ATTRIBUTE_UNUSED_ __attribute__ ((unused))
00892 #elif defined(__clang__)
00893 # if __has_attribute(unused)
00894 # define GTEST_ATTRIBUTE_UNUSED_ __attribute__ ((unused))
00895 # endif
00896 #endif
00897 #ifndef GTEST_ATTRIBUTE_UNUSED_
00898 # define GTEST_ATTRIBUTE_UNUSED_
00899 #endif
00900
00901 #if GTEST_LANG_CXX11
00902 # define GTEST_CXX11_EQUALS_DELETE_ = delete
00903 #else // GTEST_LANG_CXX11
00904 # define GTEST_CXX11_EQUALS_DELETE_
00905 #endif // GTEST_LANG_CXX11

```

```
00906 // Use this annotation before a function that takes a printf format string.
00907 #if (defined(__GNUC__) || defined(__clang__)) && !defined(COMPILER_ICC)
00908 # if defined(_MINGW_PRINTF_FORMAT)
00909 // MinGW has two different printf implementations. Ensure the format macro
00910 // matches the selected implementation. See
00911 // https://sourceforge.net/p/mingw-w64/wiki2/gnu%20printf/.
00912 # define GTEST_ATTRIBUTE_PRINTF_(string_index, first_to_check) \
00913     __attribute__((__format__(__MINGW_PRINTF_FORMAT, string_index, \
00914                             first_to_check)))
00915 # else
00916 # define GTEST_ATTRIBUTE_PRINTF_(string_index, first_to_check) \
00917     __attribute__((__format__(__printf__, string_index, first_to_check)))
00918 # endif
00919 #else
00920 #define GTEST_ATTRIBUTE_PRINTF_(string_index, first_to_check)
00921 #endif
00922 #endif
00923
00924
00925 // A macro to disallow operator=
00926 // This should be used in the private: declarations for a class.
00927 #define GTEST_DISALLOW_ASSIGN_(type) \
00928     void operator=(type const &) GTEST_CXX11_EQUALS_DELETE_
00929
00930 // A macro to disallow copy constructor and operator=
00931 // This should be used in the private: declarations for a class.
00932 #define GTEST_DISALLOW_COPY_AND_ASSIGN_(type) \
00933     type(type const &) GTEST_CXX11_EQUALS_DELETE_; \
00934     GTEST_DISALLOW_ASSIGN_(type)
00935
00936 // Tell the compiler to warn about unused return values for functions declared
00937 // with this macro. The macro should be used on function declarations
00938 // following the argument list:
00939 //
00940 //   Sprocket* AllocateSprocket() GTEST_MUST_USE_RESULT_;
00941 #if defined(__GNUC__) && (GTEST_GCC_VER_ >= 30400) && !defined(COMPILER_ICC)
00942 # define GTEST_MUST_USE_RESULT_ __attribute__ ((warn_unused_result))
00943 #else
00944 # define GTEST_MUST_USE_RESULT_
00945 #endif // __GNUC__ && (GTEST_GCC_VER_ >= 30400) && !COMPILER_ICC
00946
00947 // MS C++ compiler emits warning when a conditional expression is compile time
00948 // constant. In some contexts this warning is false positive and needs to be
00949 // suppressed. Use the following two macros in such cases:
00950 //
00951 // GTEST_INTENTIONAL_CONST_COND_PUSH_()
00952 // while (true) {
00953 //   GTEST_INTENTIONAL_CONST_COND_POP_()
00954 //}
00955 # define GTEST_INTENTIONAL_CONST_COND_PUSH_() \
00956     GTEST_DISABLE_MSC_WARNINGS_PUSH_(4127)
00957 # define GTEST_INTENTIONAL_CONST_COND_POP_() \
00958     GTEST_DISABLE_MSC_WARNINGS_POP_()
00959
00960 // Determine whether the compiler supports Microsoft's Structured Exception
00961 // Handling. This is supported by several Windows compilers but generally
00962 // does not exist on any other system.
00963 #ifndef GTEST_HAS_SEH
00964 // The user didn't tell us, so we need to figure it out.
00965
00966 # if defined(_MSC_VER) || defined(__BORLANDC__)
00967 // These two compilers are known to support SEH.
00968 # define GTEST_HAS_SEH 1
00969 # else
00970 // Assume no SEH.
00971 # define GTEST_HAS_SEH 0
00972 # endif
00973
00974 #define GTEST_IS_THREADSAFE \
00975     (GTEST_HAS_MUTEX_AND_THREAD_LOCAL_ \
00976     || (GTEST_OS_WINDOWS && !GTEST_OS_WINDOWS_PHONE && !GTEST_OS_WINDOWS_RT) \
00977     || GTEST_HAS_PTHREAD)
00978
00979 #endif // GTEST_HAS_SEH
00980
00981 // GTEST_API_ qualifies all symbols that must be exported. The definitions below
00982 // are guarded by #ifndef to give embedders a chance to define GTEST_API_ in
00983 // gtest/internal/custom/gtest-port.h
00984 #ifndef GTEST_API_
00985
00986 #ifdef _MSC_VER
00987 # if GTEST_LINKED_AS_SHARED_LIBRARY
00988 #  define GTEST_API_ __declspec(dllexport)
00989 # elif GTEST_CREATE_SHARED_LIBRARY
00990 #  define GTEST_API_ __declspec(dllexport)
00991 # endif
00992 #elif __GNUC__ >= 4 || defined(__clang__)
```

```
00993 # define GTEST_API_ __attribute__((visibility ("default")))
00994 #endif // _MSC_VER
00995
00996 #endif // GTEST_API_
00997
00998 #ifndef GTEST_API_
00999 # define GTEST_API_
01000 #endif // GTEST_API_
01001
01002 #ifndef GTEST_DEFAULT_DEATH_TEST_STYLE
01003 # define GTEST_DEFAULT_DEATH_TEST_STYLE "fast"
01004 #endif // GTEST_DEFAULT_DEATH_TEST_STYLE
01005
01006 #ifdef __GNUC__
01007 // Ask the compiler to never inline a given function.
01008 # define GTEST_NO_INLINE__ __attribute__((noinline))
01009 #else
01010 # define GTEST_NO_INLINE_
01011 #endif
01012
01013 // _LIBCPP_VERSION is defined by the libc++ library from the LLVM project.
01014 #if !defined(GTEST_HAS_CXXABI_H_)
01015 # if defined(__GLIBCXX__) || (defined(_LIBCPP_VERSION) && !defined(_MSC_VER))
01016 # define GTEST_HAS_CXXABI_H_ 1
01017 # else
01018 # define GTEST_HAS_CXXABI_H_ 0
01019 # endif
01020 #endif
01021
01022 // A function level attribute to disable checking for use of uninitialized
01023 // memory when built with MemorySanitizer.
01024 #if defined(__clang__)
01025 # if __has_feature(memory_sanitizer)
01026 # define GTEST_ATTRIBUTE_NO_SANITIZE_MEMORY_ \
01027     __attribute__((no_sanitize_memory))
01028 # else
01029 # define GTEST_ATTRIBUTE_NO_SANITIZE_MEMORY_
01030 # endif // __has_feature(memory_sanitizer)
01031 #else
01032 # define GTEST_ATTRIBUTE_NO_SANITIZE_MEMORY_
01033 #endif // __clang__
01034
01035 // A function level attribute to disable AddressSanitizer instrumentation.
01036 #if defined(__clang__)
01037 # if __has_feature(address_sanitizer)
01038 # define GTEST_ATTRIBUTE_NO_SANITIZE_ADDRESS_ \
01039     __attribute__((no_sanitize_address))
01040 # else
01041 # define GTEST_ATTRIBUTE_NO_SANITIZE_ADDRESS_
01042 # endif // __has_feature(address_sanitizer)
01043 #else
01044 # define GTEST_ATTRIBUTE_NO_SANITIZE_ADDRESS_
01045 #endif // __clang__
01046
01047 // A function level attribute to disable ThreadSanitizer instrumentation.
01048 #if defined(__clang__)
01049 # if __has_feature(thread_sanitizer)
01050 # define GTEST_ATTRIBUTE_NO_SANITIZE_THREAD_ \
01051     __attribute__((no_sanitize_thread))
01052 # else
01053 # define GTEST_ATTRIBUTE_NO_SANITIZE_THREAD_
01054 # endif // __has_feature(thread_sanitizer)
01055 #else
01056 # define GTEST_ATTRIBUTE_NO_SANITIZE_THREAD_
01057 #endif // __clang__
01058
01059 namespace testing {
01060
01061 class Message;
01062
01063 #if defined(GTEST_TUPLE_NAMESPACE_)
01064 // Import tuple and friends into the ::testing namespace.
01065 // It is part of our interface, having them in ::testing allows us to change
01066 // their types as needed.
01067 using GTEST_TUPLE_NAMESPACE_::get;
01068 using GTEST_TUPLE_NAMESPACE_::make_tuple;
01069 using GTEST_TUPLE_NAMESPACE_::tuple;
01070 using GTEST_TUPLE_NAMESPACE_::tuple_size;
01071 using GTEST_TUPLE_NAMESPACE_::tuple_element;
01072 #endif // defined(GTEST_TUPLE_NAMESPACE_)
01073
01074 namespace internal {
01075
01076 // A secret type that Google Test users don't know about. It has no
01077 // definition on purpose. Therefore it's impossible to create a
01078 // Secret object, which is what we want.
01079 class Secret;
```

```
01080 // The GTEST_COMPILE_ASSERT_ macro can be used to verify that a compile time
01081 // expression is true. For example, you could use it to verify the
01082 // size of a static array:
01083 //
01084 //
01085 //  GTEST_COMPILE_ASSERT_(GTEST_ARRAY_SIZE_(names) == NUM_NAMES,
01086 //                         names_incorrect_size);
01087 //
01088 // or to make sure a struct is smaller than a certain size:
01089 //
01090 //  GTEST_COMPILE_ASSERT_(sizeof(foo) < 128, foo_too_large);
01091 //
01092 // The second argument to the macro is the name of the variable. If
01093 // the expression is false, most compilers will issue a warning/error
01094 // containing the name of the variable.
01095
01096 #if GTEST_LANG_CXX11
01097 # define GTEST_COMPILE_ASSERT_(expr, msg) static_assert(expr, #msg)
01098 #else // !GTEST_LANG_CXX11
01099 template <bool>
01100 struct CompileAssert {
01101 };
01102
01103 # define GTEST_COMPILE_ASSERT_(expr, msg) \
01104     typedef ::testing::internal::CompileAssert<(static_cast<bool>(expr))> \
01105         msg[static_cast<bool>(expr) ? 1 : -1] GTEST_ATTRIBUTE_UNUSED_
01106 #endif // !GTEST_LANG_CXX11
01107
01108 // Implementation details of GTEST_COMPILE_ASSERT_:
01109 //
01110 // (In C++11, we simply use static_assert instead of the following)
01111 //
01112 // - GTEST_COMPILE_ASSERT_ works by defining an array type that has -1
01113 //   elements (and thus is invalid) when the expression is false.
01114 //
01115 // - The simpler definition
01116 //
01117 // #define GTEST_COMPILE_ASSERT_(expr, msg) typedef char msg[(expr) ? 1 : -1]
01118 //
01119 // does not work, as gcc supports variable-length arrays whose sizes
01120 // are determined at run-time (this is gcc's extension and not part
01121 // of the C++ standard). As a result, gcc fails to reject the
01122 // following code with the simple definition:
01123 //
01124 //     int foo;
01125 //     GTEST_COMPILE_ASSERT_(foo, msg); // not supposed to compile as foo is
01126 //                                     // not a compile-time constant.
01127 //
01128 // - By using the type CompileAssert<(bool(expr))>, we ensures that
01129 //   expr is a compile-time constant. (Template arguments must be
01130 //   determined at compile-time.)
01131 //
01132 // - The outer parentheses in CompileAssert<(bool(expr))> are necessary
01133 //   to work around a bug in gcc 3.4.4 and 4.0.1. If we had written
01134 //
01135 //     CompileAssert<bool(expr)>
01136 //
01137 // instead, these compilers will refuse to compile
01138 //
01139 //     GTEST_COMPILE_ASSERT_(5 > 0, some_message);
01140 //
01141 // (They seem to think the ">" in "5 > 0" marks the end of the
01142 // template argument list.)
01143 //
01144 // - The array size is (bool(expr) ? 1 : -1), instead of simply
01145 //
01146 //     ((expr) ? 1 : -1).
01147 //
01148 // This is to avoid running into a bug in MS VC 7.1, which
01149 // causes ((0.0) ? 1 : -1) to incorrectly evaluate to 1.
01150
01151 // StaticAssertTypeEqHelper is used by StaticAssertTypeEq defined in gtest.h.
01152 //
01153 // This template is declared, but intentionally undefined.
01154 template <typename T1, typename T2>
01155 struct StaticAssertTypeEqHelper;
01156
01157 template <typename T>
01158 struct StaticAssertTypeEqHelper<T, T> {
01159     enum { value = true };
01160 };
01161
01162 // Same as std::is_same<>.
01163 template <typename T, typename U>
01164 struct IsSame {
01165     enum { value = false };
01166 };
```

```

01167 template <typename T>
01168 struct IsSame<T, T> {
01169     enum { value = true };
01170 };
01171
01172 // Evaluates to the number of elements in 'array'.
01173 #define GTEST_ARRAY_SIZE_(array) (sizeof(array) / sizeof(array[0]))
01174
01175 #if GTEST_HAS_GLOBAL_STRING
01176 typedef ::string string;
01177 #else
01178 typedef ::std::string string;
01179 #endif // GTEST_HAS_GLOBAL_STRING
01180
01181 #if GTEST_HAS_GLOBAL_WSTRING
01182 typedef ::wstring wstring;
01183 #elif GTEST_HAS_STD_WSTRING
01184 typedef ::std::wstring wstring;
01185 #endif // GTEST_HAS_GLOBAL_WSTRING
01186
01187 // A helper for suppressing warnings on constant condition. It just
01188 // returns 'condition'.
01189 GTEST_API_ bool IsTrue(bool condition);
01190
01191 // Defines scoped_ptr.
01192
01193 // This implementation of scoped_ptr is PARTIAL - it only contains
01194 // enough stuff to satisfy Google Test's need.
01195 template <typename T>
01196 class scoped_ptr {
01197 public:
01198     typedef T element_type;
01199
01200     explicit scoped_ptr(T* p = NULL) : ptr_(p) {}
01201     ~scoped_ptr() { reset(); }
01202
01203     T& operator*() const { return *ptr_; }
01204     T* operator->() const { return ptr_; }
01205     T* get() const { return ptr_; }
01206
01207     T* release() {
01208         T* const ptr = ptr_;
01209         ptr_ = NULL;
01210         return ptr;
01211     }
01212
01213     void reset(T* p = NULL) {
01214         if (p != ptr_) {
01215             if (IsTrue(sizeof(T) > 0)) { // Makes sure T is a complete type.
01216                 delete ptr_;
01217             }
01218             ptr_ = p;
01219         }
01220     }
01221
01222     friend void swap(scoped_ptr& a, scoped_ptr& b) {
01223         using std::swap;
01224         swap(a.ptr_, b.ptr_);
01225     }
01226
01227 private:
01228     T* ptr_;
01229
01230     GTEST_DISALLOW_COPY_AND_ASSIGN_(scoped_ptr);
01231 };
01232
01233 // Defines RE.
01234
01235 #if GTEST_USES_PCRE
01236 // If used, PCRE is injected by custom/gtest-port.h
01237 #elif GTEST_USES_POSIX_RE || GTEST_USES_SIMPLE_RE
01238
01239 // A simple C++ wrapper for <regex.h>. It uses the POSIX Extended
01240 // Regular Expression syntax.
01241 class GTEST_API_ RE {
01242 public:
01243     // A copy constructor is required by the Standard to initialize object
01244     // references from r-values.
01245     RE(const RE& other) { Init(other.pattern()); }
01246
01247     // Constructs an RE from a string.
01248     RE(const ::std::string& regex) { Init(regex.c_str()); } // NOLINT
01249
01250 #if GTEST_HAS_GLOBAL_STRING
01251     RE(const ::string& regex) { Init(regex.c_str()); } // NOLINT
01252

```

```
01254 # endif // GTEST_HAS_GLOBAL_STRING
01255
01256     RE(const char* regex) { Init(regex); } // NOLINT
01257     ~RE();
01258
01259     // Returns the string representation of the regex.
01260     const char* pattern() const { return pattern_; }
01261
01262     // FullMatch(str, re) returns true iff regular expression re matches
01263     // the entire str.
01264     // PartialMatch(str, re) returns true iff regular expression re
01265     // matches a substring of str (including str itself).
01266     //
01267     // FIXME: make FullMatch() and PartialMatch() work
01268     // when str contains NUL characters.
01269     static bool FullMatch(const ::std::string& str, const RE& re) {
01270         return FullMatch(str.c_str(), re);
01271     }
01272     static bool PartialMatch(const ::std::string& str, const RE& re) {
01273         return PartialMatch(str.c_str(), re);
01274     }
01275
01276 # if GTEST_HAS_GLOBAL_STRING
01277
01278     static bool FullMatch(const ::string& str, const RE& re) {
01279         return FullMatch(str.c_str(), re);
01280     }
01281     static bool PartialMatch(const ::string& str, const RE& re) {
01282         return PartialMatch(str.c_str(), re);
01283     }
01284
01285 # endif // GTEST_HAS_GLOBAL_STRING
01286
01287     static bool FullMatch(const char* str, const RE& re);
01288     static bool PartialMatch(const char* str, const RE& re);
01289
01290 private:
01291     void Init(const char* regex);
01292
01293     // We use a const char* instead of an std::string, as Google Test used to be
01294     // used where std::string is not available. FIXME: change to
01295     // std::string.
01296     const char* pattern_;
01297     bool is_valid_;
01298
01299 # if GTEST_USES_POSIX_REGEX
01300
01301     regex_t full_regex_; // For FullMatch().
01302     regex_t partial_regex_; // For PartialMatch().
01303
01304 # else // GTEST_USES_SIMPLE_REGEX
01305
01306     const char* full_pattern_; // For FullMatch();
01307
01308 # endif
01309
01310     GTEST_DISALLOW_ASSIGN_(RE);
01311 };
01312
01313 #endif // GTEST_USES_PCRE
01314
01315 // Formats a source file path and a line number as they would appear
01316 // in an error message from the compiler used to compile this code.
01317 GTEST_API_ ::std::string FormatFileLocation(const char* file, int line);
01318
01319 // Formats a file location for compiler-independent XML output.
01320 // Although this function is not platform dependent, we put it next to
01321 // FormatFileLocation in order to contrast the two functions.
01322 GTEST_API_ ::std::string FormatCompilerIndependentFileLocation(const char* file,
01323                                         int line);
01324
01325 // Defines logging utilities:
01326 //   GTEST_LOG_(severity) - logs messages at the specified severity level. The
01327 //                         message itself is streamed into the macro.
01328 //   LogToStderr() - directs all log messages to stderr.
01329 //   FlushInfoLog() - flushes informational log messages.
01330
01331 enum GTestLogSeverity {
01332     GTEST_INFO,
01333     GTEST_WARNING,
01334     GTEST_ERROR,
01335     GTEST_FATAL
01336 };
01337
01338 // Formats log entry severity, provides a stream object for streaming the
01339 // log message, and terminates the message with a newline when going out of
01340 // scope.
```

```

01341 class GTEST_API_ GTestLog {
01342 public:
01343     GTestLog(GTestLogSeverity severity, const char* file, int line);
01344
01345     // Flushes the buffers and, if severity is GTEST_FATAL, aborts the program.
01346     ~GTestLog();
01347
01348     ::std::ostream& GetStream() { return ::std::cerr; }
01349
01350 private:
01351     const GTestLogSeverity severity_;
01352
01353     GTEST_DISALLOW_COPY_AND_ASSIGN_(GTestLog);
01354 };
01355
01356 #if !defined(GTEST_LOG_)
01357
01358 # define GTEST_LOG_(severity) \
01359     ::testing::internal::GTestLog(::testing::internal::GTEST_##severity, \
01360                                     __FILE__, __LINE__).GetStream()
01361
01362 inline void LogToStderr() {}
01363 inline void FlushInfoLog() { fflush(NULL); }
01364
01365 #endif // !defined(GTEST_LOG_)
01366
01367 #if !defined(GTEST_CHECK_)
01368 // INTERNAL IMPLEMENTATION - DO NOT USE.
01369 //
01370 // GTEST_CHECK_ is an all-mode assert. It aborts the program if the condition
01371 // is not satisfied.
01372 // Synopsys:
01373 //     GTEST_CHECK_(boolean_condition);
01374 //     or
01375 //     GTEST_CHECK_(boolean_condition) « "Additional message";
01376 //
01377 // This checks the condition and if the condition is not satisfied
01378 // it prints message about the condition violation, including the
01379 // condition itself, plus additional message streamed into it, if any,
01380 // and then it aborts the program. It aborts the program irrespective of
01381 // whether it is built in the debug mode or not.
01382 # define GTEST_CHECK_(condition) \
01383     GTEST_AMBIGUOUS_ELSE_BLOCKER_ \
01384     if (::testing::internal::IsTrue(condition)) \
01385     ; \
01386     else \
01387         GTEST_LOG_(FATAL) « "Condition " #condition " failed. "
01388 #endif // !defined(GTEST_CHECK_)
01389
01390 // An all-mode assert to verify that the given POSIX-style function
01391 // call returns 0 (indicating success). Known limitation: this
01392 // doesn't expand to a balanced 'if' statement, so enclose the macro
01393 // in {} if you need to use it as the only statement in an 'if'
01394 // branch.
01395 #define GTEST_CHECK_POSIX_SUCCESS_(posix_call) \
01396     if (const int gtest_error = (posix_call)) \
01397         GTEST_LOG_(FATAL) « #posix_call « "failed with error " \
01398             « gtest_error
01399
01400 // Adds reference to a type if it is not a reference type,
01401 // otherwise leaves it unchanged. This is the same as
01402 // tr1::add_reference, which is not widely available yet.
01403 template <typename T>
01404 struct AddReference { typedef T& type; }; // NOLINT
01405 template <typename T>
01406 struct AddReference<T> { typedef T& type; }; // NOLINT
01407
01408 // A handy wrapper around AddReference that works when the argument T
01409 // depends on template parameters.
01410 #define GTEST_ADD_REFERENCE_(T) \
01411     typename ::testing::internal::AddReference<T>::type
01412
01413 // Transforms "T" into "const T&" according to standard reference collapsing
01414 // rules (this is only needed as a backport for C++98 compilers that do not
01415 // support reference collapsing). Specifically, it transforms:
01416 //
01417 //     char      ==> const char&
01418 //     const char ==> const char&
01419 //     char&    ==> char&
01420 //     const char& ==> const char&
01421 //
01422 // Note that the non-const reference will not have "const" added. This is
01423 // standard, and necessary so that "T" can always bind to "const T&".
01424 template <typename T>
01425 struct ConstRef { typedef const T& type; };
01426 template <typename T>
01427 struct ConstRef<T> { typedef T& type; };

```

```

01428
01429 // The argument T must depend on some template parameters.
01430 #define GTEST_REFERENCE_TO_CONST_(T) \
01431     typename ::testing::internal::ConstRef<T>::type
01432
01433 #if GTEST_HAS_STD_MOVE_
01434 using std::forward;
01435 using std::move;
01436
01437 template <typename T>
01438 struct RvalueRef {
01439     typedef T& type;
01440 };
01441 #else // GTEST_HAS_STD_MOVE_
01442 template <typename T>
01443 const T& move(const T& t) {
01444     return t;
01445 }
01446 template <typename T>
01447 GTEST_ADD_REFERENCE_(T) forward(GTEST_ADD_REFERENCE_(T) t) { return t; }
01448
01449 template <typename T>
01450 struct RvalueRef {
01451     typedef const T& type;
01452 };
01453 #endif // GTEST_HAS_STD_MOVE_
01454
01455 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
01456 //
01457 // Use ImplicitCast_ as a safe version of static_cast for upcasting in
01458 // the type hierarchy (e.g. casting a Foo* to a SuperclassOfFoo* or a
01459 // const Foo*). When you use ImplicitCast_, the compiler checks that
01460 // the cast is safe. Such explicit ImplicitCast_s are necessary in
01461 // surprisingly many situations where C++ demands an exact type match
01462 // instead of an argument type convertible to a target type.
01463 //
01464 // The syntax for using ImplicitCast_ is the same as for static_cast:
01465 //
01466 //     ImplicitCast_<ToType>(expr)
01467 //
01468 // ImplicitCast_ would have been part of the C++ standard library,
01469 // but the proposal was submitted too late. It will probably make
01470 // its way into the language in the future.
01471 //
01472 // This relatively ugly name is intentional. It prevents clashes with
01473 // similar functions users may have (e.g., implicit_cast). The internal
01474 // namespace alone is not enough because the function can be found by ADL.
01475 template<typename To>
01476 inline To ImplicitCast_(To x) { return x; }
01477
01478 // When you upcast (that is, cast a pointer from type Foo to type
01479 // SuperclassOfFoo), it's fine to use ImplicitCast_<>, since upcasts
01480 // always succeed. When you downcast (that is, cast a pointer from
01481 // type Foo to type SubclassOfFoo), static_cast<> isn't safe, because
01482 // how do you know the pointer is really of type SubclassOfFoo? It
01483 // could be a bare Foo, or of type DifferentSubclassOfFoo. Thus,
01484 // when you downcast, you should use this macro. In debug mode, we
01485 // use dynamic_cast<> to double-check the downcast is legal (we die
01486 // if it's not). In normal mode, we do the efficient static_cast<>
01487 // instead. Thus, it's important to test in debug mode to make sure
01488 // the cast is legal!
01489 //     This is the only place in the code we should use dynamic_cast<>.
01490 // In particular, you SHOULDN'T be using dynamic_cast<> in order to
01491 // do RTTI (eg code like this:
01492 //     if (dynamic_cast<Subclass1>(foo)) HandleASubclass1Object(foo);
01493 //     if (dynamic_cast<Subclass2>(foo)) HandleASubclass2Object(foo);
01494 // You should design the code some other way not to need this.
01495 //
01496 // This relatively ugly name is intentional. It prevents clashes with
01497 // similar functions users may have (e.g., down_cast). The internal
01498 // namespace alone is not enough because the function can be found by ADL.
01499 template<typename To, typename From> // use like this: DownCast_-<T*>(foo);
01500 inline To DownCast_(From* f) { // so we only accept pointers
01501     // Ensures that To is a sub-type of From *. This test is here only
01502     // for compile-time type checking, and has no overhead in an
01503     // optimized build at run-time, as it will be optimized away
01504     // completely.
01505     GTEST_INTENTIONAL_CONST_COND_PUSH_()
01506     if (false) {
01507         GTEST_INTENTIONAL_CONST_COND_POP_()
01508         const To to = NULL;
01509         ::testing::internal::ImplicitCast_-<From*>(to);
01510     }
01511
01512 #if GTEST_HAS_RTTI
01513 // RTTI: debug mode only!
01514     GTEST_CHECK_(f == NULL || dynamic_cast<To>(f) != NULL);

```

```

01515 #endif
01516     return static_cast<To>(f);
01517 }
01518
01519 // Downcasts the pointer of type Base to Derived.
01520 // Derived must be a subclass of Base. The parameter MUST
01521 // point to a class of type Derived, not any subclass of it.
01522 // When RTTI is available, the function performs a runtime
01523 // check to enforce this.
01524 template <class Derived, class Base>
01525 Derived* CheckedDowncastToActualType(Base* base) {
01526 #if GTEST_HAS_RTTI
01527     GTEST_CHECK_(typeid(*base) == typeid(Derived));
01528 #endif
01529
01530 #if GTEST_HAS_DOWNCAST_
01531     return ::down_cast<Derived*>(base);
01532 #elif GTEST_HAS_RTTI
01533     return dynamic_cast<Derived*>(base); // NOLINT
01534 #else
01535     return static_cast<Derived*>(base); // Poor man's downcast.
01536 #endif
01537 }
01538
01539 #if GTEST_HAS_STREAM_REDIRECTION
01540
01541 // Defines the stderr capturer:
01542 // CaptureStdout - starts capturing stdout.
01543 // GetCapturedStdout - stops capturing stdout and returns the captured string.
01544 // CaptureStderr - starts capturing stderr.
01545 // GetCapturedStderr - stops capturing stderr and returns the captured string.
01546 //
01547 GTEST_API_ void CaptureStdout();
01548 GTEST_API_ std::string GetCapturedStdout();
01549 GTEST_API_ void CaptureStderr();
01550 GTEST_API_ std::string GetCapturedStderr();
01551
01552 #endif // GTEST_HAS_STREAM_REDIRECTION
01553 // Returns the size (in bytes) of a file.
01554 GTEST_API_ size_t GetFileSize(FILE* file);
01555
01556 // Reads the entire content of a file as a string.
01557 GTEST_API_ std::string ReadEntireFile(FILE* file);
01558
01559 // All command line arguments.
01560 GTEST_API_ std::vector<std::string> GetArgs();
01561
01562 #if GTEST_HAS_DEATH_TEST
01563
01564 std::vector<std::string> GetInjectableArgs();
01565 // Deprecated: pass the args vector by value instead.
01566 void SetInjectableArgs(const std::vector<std::string>* new_args);
01567 void SetInjectableArgs(const std::vector<std::string>& new_args);
01568 #if GTEST_HAS_GLOBAL_STRING
01569 void SetInjectableArgs(const std::vector<::string>& new_args);
01570 #endif // GTEST_HAS_GLOBAL_STRING
01571 void ClearInjectableArgs();
01572
01573 #endif // GTEST_HAS_DEATH_TEST
01574
01575 // Defines synchronization primitives.
01576 #if GTEST_IS_THREADSAFE
01577 # if GTEST_HAS_PTHREAD
01578 // Sleeps for (roughly) n milliseconds. This function is only for testing
01579 // Google Test's own constructs. Don't use it in user tests, either
01580 // directly or indirectly.
01581 inline void SleepMilliseconds(int n) {
01582     const timespec time = {
01583         0, // 0 seconds.
01584         n * 1000L * 1000L, // And n ms.
01585     };
01586     nanosleep(&time, NULL);
01587 }
01588 # endif // GTEST_HAS_PTHREAD
01589
01590 # if GTEST_HAS_NOTIFICATION_
01591 // Notification has already been imported into the namespace.
01592 // Nothing to do here.
01593
01594 # elif GTEST_HAS_PTHREAD
01595 // Allows a controller thread to pause execution of newly created
01596 // threads until notified. Instances of this class must be created
01597 // and destroyed in the controller thread.
01598 //
01599 // This class is only for testing Google Test's own constructs. Do not
01600 // use it in user tests, either directly or indirectly.
01601 class Notification {

```

```
01602 public:
01603 Notification() : notified_(false) {
01604     GTEST_CHECK_POSIX_SUCCESS_(pthread_mutex_init(&mutex_, NULL));
01605 }
01606 ~Notification() {
01607     pthread_mutex_destroy(&mutex_);
01608 }
01609 // Notifies all threads created with this notification to start. Must
01610 // be called from the controller thread.
01611 void Notify() {
01612     pthread_mutex_lock(&mutex_);
01613     notified_ = true;
01614     pthread_mutex_unlock(&mutex_);
01615 }
01616
01617 // Blocks until the controller thread notifies. Must be called from a test
01618 // thread.
01619 void WaitForNotification() {
01620     for (;;) {
01621         pthread_mutex_lock(&mutex_);
01622         const bool notified = notified_;
01623         pthread_mutex_unlock(&mutex_);
01624         if (notified)
01625             break;
01626         SleepMilliseconds(10);
01627     }
01628 }
01629 }
01630
01631 private:
01632     pthread_mutex_t mutex_;
01633     bool notified_;
01634
01635     GTEST_DISALLOW_COPY_AND_ASSIGN_(Notification);
01636 };
01637
01638 # elif GTEST_OS_WINDOWS && !GTEST_OS_WINDOWS_PHONE && !GTEST_OS_WINDOWS_RT
01639
01640 GTEST_API_ void SleepMilliseconds(int n);
01641
01642 // Provides leak-safe Windows kernel handle ownership.
01643 // Used in death tests and in threading support.
01644 class GTEST_API_ AutoHandle {
01645 public:
01646     // Assume that Win32 HANDLE type is equivalent to void*. Doing so allows us to
01647     // avoid including <windows.h> in this header file. Including <windows.h> is
01648     // undesirable because it defines a lot of symbols and macros that tend to
01649     // conflict with client code. This assumption is verified by
01650     // WindowsTypesTest.HANDLEIsVoidStar.
01651     typedef void* Handle;
01652     AutoHandle();
01653     explicit AutoHandle(Handle handle);
01654
01655     ~AutoHandle();
01656
01657     Handle Get() const;
01658     void Reset();
01659     void Reset(Handle handle);
01660
01661 private:
01662     // Returns true iff the handle is a valid handle object that can be closed.
01663     bool IsCloseable() const;
01664
01665     Handle handle_;
01666
01667     GTEST_DISALLOW_COPY_AND_ASSIGN_(AutoHandle);
01668 };
01669
01670 // Allows a controller thread to pause execution of newly created
01671 // threads until notified. Instances of this class must be created
01672 // and destroyed in the controller thread.
01673 //
01674 // This class is only for testing Google Test's own constructs. Do not
01675 // use it in user tests, either directly or indirectly.
01676 class GTEST_API_ Notification {
01677 public:
01678     Notification();
01679     void Notify();
01680     void WaitForNotification();
01681
01682 private:
01683     AutoHandle event_;
01684
01685     GTEST_DISALLOW_COPY_AND_ASSIGN_(Notification);
01686 };
01687 # endif // GTEST_HAS_NOTIFICATION_
01688
```

```

01689 // On MinGW, we can have both GTEST_OS_WINDOWS and GTEST_HAS_PTHREAD
01690 // defined, but we don't want to use MinGW's pthreads implementation, which
01691 // has conformance problems with some versions of the POSIX standard.
01692 # if GTEST_HAS_PTHREAD && !GTEST_OS_WINDOWS_MINGW
01693
01694 // As a C-function, ThreadFuncWithCLinkage cannot be templated itself.
01695 // Consequently, it cannot select a correct instantiation of ThreadWithParam
01696 // in order to call its Run(). Introducing ThreadWithParamBase as a
01697 // non-templated base class for ThreadWithParam allows us to bypass this
01698 // problem.
01699 class ThreadWithParamBase {
01700 public:
01701     virtual ~ThreadWithParamBase() {}
01702     virtual void Run() = 0;
01703 };
01704
01705 // pthread_create() accepts a pointer to a function type with the C linkage.
01706 // According to the Standard (7.5/1), function types with different linkages
01707 // are different even if they are otherwise identical. Some compilers (for
01708 // example, SunStudio) treat them as different types. Since class methods
01709 // cannot be defined with C-linkage we need to define a free C-function to
01710 // pass into pthread_create().
01711 extern "C" inline void* ThreadFuncWithCLinkage(void* thread) {
01712     static_cast<ThreadWithParamBase*>(thread)->Run();
01713     return NULL;
01714 }
01715
01716 // Helper class for testing Google Test's multi-threading constructs.
01717 // To use it, write:
01718 //
01719 //     void ThreadFunc(int param) { /* Do things with param */ }
01720 //     Notification thread_can_start;
01721 //     ...
01722 //     // The thread_can_start parameter is optional; you can supply NULL.
01723 //     ThreadWithParam<int> thread(&ThreadFunc, 5, &thread_can_start);
01724 //     thread_can_start.Notify();
01725 //
01726 // These classes are only for testing Google Test's own constructs. Do
01727 // not use them in user tests, either directly or indirectly.
01728 template <typename T>
01729 class ThreadWithParam : public ThreadWithParamBase {
01730 public:
01731     typedef void UserThreadFunc(T);
01732
01733     ThreadWithParam(UserThreadFunc* func, T param, Notification* thread_can_start)
01734         : func_(func),
01735             param_(param),
01736             thread_can_start_(thread_can_start),
01737             finished_(false) {
01738         ThreadWithParamBase* const base = this;
01739         // The thread can be created only after all fields except thread_
01740         // have been initialized.
01741         GTEST_CHECK_POSIX_SUCCESS(
01742             pthread_create(&thread_, 0, &ThreadFuncWithCLinkage, base));
01743     }
01744     ~ThreadWithParam() { Join(); }
01745
01746     void Join() {
01747         if (!finished_) {
01748             GTEST_CHECK_POSIX_SUCCESS(pthread_join(thread_, 0));
01749             finished_ = true;
01750         }
01751     }
01752
01753     virtual void Run() {
01754         if (thread_can_start_ != NULL)
01755             thread_can_start_->WaitForNotification();
01756         func_(param_);
01757     }
01758
01759 private:
01760     UserThreadFunc* const func_; // User-supplied thread function.
01761     const T param_; // User-supplied parameter to the thread function.
01762     // When non-NULL, used to block execution until the controller thread
01763     // notifies.
01764     Notification const thread_can_start_;
01765     bool finished_; // true iff we know that the thread function has finished.
01766     pthread_t thread_; // The native thread object.
01767
01768     GTEST_DISALLOW_COPY_AND_ASSIGN(ThreadWithParam);
01769 };
01770 # endif // !GTEST_OS_WINDOWS && GTEST_HAS_PTHREAD ||
01771 // GTEST_HAS_MUTEX_AND_THREAD_LOCAL_
01772
01773 # if GTEST_HAS_MUTEX_AND_THREAD_LOCAL_
01774 // Mutex and ThreadLocal have already been imported into the namespace.
01775 // Nothing to do here.

```

```
01776
01777 # elif GTEST_OS_WINDOWS && !GTEST_OS_WINDOWS_PHONE && !GTEST_OS_WINDOWS_RT
01778
01779 // Mutex implements mutex on Windows platforms. It is used in conjunction
01780 // with class MutexLock:
01781 //
01782 //   Mutex mutex;
01783 // ...
01784 //   MutexLock lock(&mutex); // Acquires the mutex and releases it at the
01785 //                           // end of the current scope.
01786 //
01787 // A static Mutex *must* be defined or declared using one of the following
01788 // macros:
01789 //   GTEST_DEFINE_STATIC_MUTEX_(g_some_mutex);
01790 //   GTEST_DECLARE_STATIC_MUTEX_(g_some_mutex);
01791 //
01792 // (A non-static Mutex is defined/declared in the usual way).
01793 class GTEST_API_ Mutex {
01794 public:
01795     enum MutexType { kStatic = 0, kDynamic = 1 };
01796     // We rely on kStaticMutex being 0 as it is to what the linker initializes
01797     // type_ in static mutexes. critical_section_ will be initialized lazily
01798     // in ThreadSafeLazyInit().
01799     enum StaticConstructorSelector { kStaticMutex = 0 };
01800
01801     // This constructor intentionally does nothing. It relies on type_ being
01802     // statically initialized to 0 (effectively setting it to kStatic) and on
01803     // ThreadSafeLazyInit() to lazily initialize the rest of the members.
01804     explicit Mutex(StaticConstructorSelector /*dummy*/) {}
01805
01806     Mutex();
01807     ~Mutex();
01808
01809     void Lock();
01810
01811     void Unlock();
01812
01813     // Does nothing if the current thread holds the mutex. Otherwise, crashes
01814     // with high probability.
01815     void AssertHeld();
01816
01817 private:
01818     // Initializes owner_thread_id_ and critical_section_ in static mutexes.
01819     void ThreadSafeLazyInit();
01820
01821     // Per https://blogs.microsoft.com/oldnewthing/20040223-00/?p=40503,
01822     // we assume that 0 is an invalid value for thread IDs.
01823     unsigned int owner_thread_id_;
01824
01825     // For static mutexes, we rely on these members being initialized to zeros
01826     // by the linker.
01827     MutexType type_;
01828     long critical_section_init_phase_; // NOLINT
01829     GTEST_CRITICAL_SECTION* critical_section_;
01830
01831     GTEST_DISALLOW_COPY_AND_ASSIGN_(Mutex);
01832 };
01833
01834 # define GTEST_DECLARE_STATIC_MUTEX_(mutex) \
01835     extern ::testing::internal::Mutex mutex
01836
01837 # define GTEST_DEFINE_STATIC_MUTEX_(mutex) \
01838     ::testing::internal::Mutex mutex(::testing::internal::Mutex::kStaticMutex)
01839
01840 // We cannot name this class MutexLock because the ctor declaration would
01841 // conflict with a macro named MutexLock, which is defined on some
01842 // platforms. That macro is used as a defensive measure to prevent against
01843 // inadvertent misuses of MutexLock like "MutexLock(&mu)" rather than
01844 // "MutexLock l(&mu)". Hence the typedef trick below.
01845 class GTestMutexLock {
01846 public:
01847     explicit GTestMutexLock(Mutex* mutex)
01848         : mutex_(mutex) { mutex_->Lock(); }
01849
01850     ~GTestMutexLock() { mutex_->Unlock(); }
01851
01852 private:
01853     Mutex* const mutex_;
01854
01855     GTEST_DISALLOW_COPY_AND_ASSIGN_(GTestMutexLock);
01856 };
01857
01858 typedef GTestMutexLock MutexLock;
01859
01860 // Base class for ValueHolder<T>. Allows a caller to hold and delete a value
01861 // without knowing its type.
01862 class ThreadLocalValueHolderBase {
```

```
01863 public:
01864     virtual ~ThreadLocalValueHolderBase() {}
01865 };
01866
01867 // Provides a way for a thread to send notifications to a ThreadLocal
01868 // regardless of its parameter type.
01869 class ThreadLocalBase {
01870 public:
01871     // Creates a new ValueHolder<T> object holding a default value passed to
01872     // this ThreadLocal<T>'s constructor and returns it. It is the caller's
01873     // responsibility not to call this when the ThreadLocal<T> instance already
01874     // has a value on the current thread.
01875     virtual ThreadLocalValueHolderBase* NewValueForCurrentThread() const = 0;
01876
01877 protected:
01878     ThreadLocalBase() {}
01879     virtual ~ThreadLocalBase() {}
01880
01881 private:
01882     GTEST_DISALLOW_COPY_AND_ASSIGN_(ThreadLocalBase);
01883 };
01884
01885 // Maps a thread to a set of ThreadLocals that have values instantiated on that
01886 // thread and notifies them when the thread exits. A ThreadLocal instance is
01887 // expected to persist until all threads it has values on have terminated.
01888 class GTEST_API_ ThreadLocalRegistry {
01889 public:
01890     // Registers thread_local_instance as having value on the current thread.
01891     // Returns a value that can be used to identify the thread from other threads.
01892     static ThreadLocalValueHolderBase* GetValueOnCurrentThread(
01893         const ThreadLocalBase* thread_local_instance);
01894
01895     // Invoked when a ThreadLocal instance is destroyed.
01896     static void OnThreadLocalDestroyed(
01897         const ThreadLocalBase* thread_local_instance);
01898 };
01899
01900 class GTEST_API_ ThreadWithParamBase {
01901 public:
01902     void Join();
01903
01904 protected:
01905     class Runnable {
01906 public:
01907         virtual ~Runnable() {}
01908         virtual void Run() = 0;
01909     };
01910
01911     ThreadWithParamBase(Runnable *runnable, Notification* thread_can_start);
01912     virtual ~ThreadWithParamBase();
01913
01914 private:
01915     AutoHandle thread_;
01916 };
01917
01918 // Helper class for testing Google Test's multi-threading constructs.
01919 template <typename T>
01920 class ThreadWithParam : public ThreadWithParamBase {
01921 public:
01922     typedef void UserThreadFunc(T);
01923
01924     ThreadWithParam(UserThreadFunc* func, T param, Notification* thread_can_start)
01925         : ThreadWithParamBase(new RunnableImpl(func, param), thread_can_start) {
01926     }
01927     virtual ~ThreadWithParam() {}
01928
01929 private:
01930     class RunnableImpl : public Runnable {
01931 public:
01932         RunnableImpl(UserThreadFunc* func, T param)
01933             : func_(func),
01934                 param_(param) {
01935         }
01936         virtual ~RunnableImpl() {}
01937         virtual void Run() {
01938             func_(param_);
01939         }
01940
01941 private:
01942     UserThreadFunc* const func_;
01943     const T param_;
01944
01945     GTEST_DISALLOW_COPY_AND_ASSIGN_(RunnableImpl);
01946 };
01947
01948     GTEST_DISALLOW_COPY_AND_ASSIGN_(ThreadWithParam);
01949 };
```

```
01950 // Implements thread-local storage on Windows systems.
01951 //
01952 //
01953 //   // Thread 1
01954 //   ThreadLocal<int> tl(100); // 100 is the default value for each thread.
01955 //
01956 //   // Thread 2
01957 //   tl.set(150); // Changes the value for thread 2 only.
01958 //   EXPECT_EQ(150, tl.get());
01959 //
01960 //   // Thread 1
01961 //   EXPECT_EQ(100, tl.get()); // In thread 1, tl has the original value.
01962 //   tl.set(200);
01963 //   EXPECT_EQ(200, tl.get());
01964 //
01965 // The template type argument T must have a public copy constructor.
01966 // In addition, the default ThreadLocal constructor requires T to have
01967 // a public default constructor.
01968 //
01969 // The users of a ThreadLocal instance have to make sure that all but one
01970 // threads (including the main one) using that instance have exited before
01971 // destroying it. Otherwise, the per-thread objects managed for them by the
01972 // ThreadLocal instance are not guaranteed to be destroyed on all platforms.
01973 //
01974 // Google Test only uses global ThreadLocal objects. That means they
01975 // will die after main() has returned. Therefore, no per-thread
01976 // object managed by Google Test will be leaked as long as all threads
01977 // using Google Test have exited when main() returns.
01978 template <typename T>
01979 class ThreadLocal : public ThreadLocalBase {
01980 public:
01981     ThreadLocal() : default_factory_(new DefaultValueHolderFactory()) {}
01982     explicit ThreadLocal(const T& value)
01983         : default_factory_(new InstanceValueHolderFactory(value)) {}
01984
01985     ~ThreadLocal() { ThreadLocalRegistry::OnThreadLocalDestroyed(this); }
01986
01987     T* pointer() { return GetOrCreateValue(); }
01988     const T* pointer() const { return GetOrCreateValue(); }
01989     const T& get() const { return *pointer(); }
01990     void set(const T& value) { *pointer() = value; }
01991
01992 private:
01993     // Holds a value of T. Can be deleted via its base class without the caller
01994     // knowing the type of T.
01995     class ValueHolder : public ThreadLocalValueHolderBase {
01996     public:
01997         ValueHolder() : value_() {}
01998         explicit ValueHolder(const T& value) : value_(value) {}
01999
02000         T* pointer() { return &value_; }
02001
02002     private:
02003         T value_;
02004         GTEST_DISALLOW_COPY_AND_ASSIGN_(ValueHolder);
02005     };
02006
02007
02008     T* GetOrCreateValue() const {
02009         return static_cast<ValueHolder*>(
02010             ThreadLocalRegistry::GetValueOnCurrentThread(this))->pointer();
02011     }
02012
02013     virtual ThreadLocalValueHolderBase* NewValueForCurrentThread() const {
02014         return default_factory_->MakeNewHolder();
02015     }
02016
02017     class ValueHolderFactory {
02018     public:
02019         ValueHolderFactory() {}
02020         virtual ~ValueHolderFactory() {}
02021         virtual ValueHolder* MakeNewHolder() const = 0;
02022
02023     private:
02024         GTEST_DISALLOW_COPY_AND_ASSIGN_(ValueHolderFactory);
02025     };
02026
02027     class DefaultValueHolderFactory : public ValueHolderFactory {
02028     public:
02029         DefaultValueHolderFactory() {}
02030         virtual ValueHolder* MakeNewHolder() const { return new ValueHolder(); }
02031
02032     private:
02033         GTEST_DISALLOW_COPY_AND_ASSIGN_(DefaultValueHolderFactory);
02034     };
02035
02036     class InstanceValueHolderFactory : public ValueHolderFactory {
```

```

02037     public:
02038         explicit InstanceValueHolderFactory(const T& value) : value_(value) {}
02039         virtual ValueHolder* MakeNewHolder() const {
02040             return new ValueHolder(value_);
02041         }
02042
02043     private:
02044         const T value_; // The value for each thread.
02045
02046         GTEST_DISALLOW_COPY_AND_ASSIGN_(InstanceValueHolderFactory);
02047     };
02048
02049     scoped_ptr<ValueHolderFactory> default_factory_;
02050
02051     GTEST_DISALLOW_COPY_AND_ASSIGN_(ThreadLocal);
02052 };
02053
02054 # elif GTEST_HAS_PTHREAD
02055
02056 // MutexBase and Mutex implement mutex on pthreads-based platforms.
02057 class MutexBase {
02058     public:
02059         // Acquires this mutex.
02060         void Lock() {
02061             GTEST_CHECK_POSIX_SUCCESS_(pthread_mutex_lock(&mutex_));
02062             owner_ = pthread_self();
02063             has_owner_ = true;
02064         }
02065
02066         // Releases this mutex.
02067         void Unlock() {
02068             // Since the lock is being released the owner_ field should no longer be
02069             // considered valid. We don't protect writing to has_owner_ here, as it's
02070             // the caller's responsibility to ensure that the current thread holds the
02071             // mutex when this is called.
02072             has_owner_ = false;
02073             GTEST_CHECK_POSIX_SUCCESS_(pthread_mutex_unlock(&mutex_));
02074         }
02075
02076         // Does nothing if the current thread holds the mutex. Otherwise, crashes
02077         // with high probability.
02078         void AssertHeld() const {
02079             GTEST_CHECK_(has_owner_ && pthread_equal(owner_, pthread_self()))
02080                 << "The current thread is not holding the mutex @" << this;
02081         }
02082
02083         // A static mutex may be used before main() is entered. It may even
02084         // be used before the dynamic initialization stage. Therefore we
02085         // must be able to initialize a static mutex object at link time.
02086         // This means MutexBase has to be a POD and its member variables
02087         // have to be public.
02088     public:
02089         pthread_mutex_t mutex_; // The underlying pthread mutex.
02090         // has_owner_ indicates whether the owner_ field below contains a valid thread
02091         // ID and is therefore safe to inspect (e.g., to use in pthread_equal()). All
02092         // accesses to the owner_ field should be protected by a check of this field.
02093         // An alternative might be to memset() owner_ to all zeros, but there's no
02094         // guarantee that a zero'd pthread_t is necessarily invalid or even different
02095         // from pthread_self().
02096         bool has_owner_;
02097         pthread_t owner_; // The thread holding the mutex.
02098     };
02099
02100 // Forward-declares a static mutex.
02101 # define GTEST_DECLARE_STATIC_MUTEX_(mutex) \
02102     extern ::testing::internal::MutexBase mutex
02103
02104 // Defines and statically (i.e. at link time) initializes a static mutex.
02105 // The initialization list here does not explicitly initialize each field,
02106 // instead relying on default initialization for the unspecified fields. In
02107 // particular, the owner_ field (a pthread_t) is not explicitly initialized.
02108 // This allows initialization to work whether pthread_t is a scalar or struct.
02109 // The flag -Wmissing-field-initializers must not be specified for this to work.
02110 #define GTEST_DEFINE_STATIC_MUTEX_(mutex) \
02111     ::testing::internal::MutexBase mutex = {PTHREAD_MUTEX_INITIALIZER, false, 0}
02112
02113 // The Mutex class can only be used for mutexes created at runtime. It
02114 // shares its API with MutexBase otherwise.
02115 class Mutex : public MutexBase {
02116     public:
02117         Mutex() {
02118             GTEST_CHECK_POSIX_SUCCESS_(pthread_mutex_init(&mutex_, NULL));
02119             has_owner_ = false;
02120         }
02121         ~Mutex() {
02122             GTEST_CHECK_POSIX_SUCCESS_(pthread_mutex_destroy(&mutex_));
02123         }

```

```
02124
02125     private:
02126         GTEST_DISALLOW_COPY_AND_ASSIGN_(Mutex);
02127     };
02128
02129     // We cannot name this class MutexLock because the ctor declaration would
02130     // conflict with a macro named MutexLock, which is defined on some
02131     // platforms. That macro is used as a defensive measure to prevent against
02132     // inadvertent misuses of MutexLock like "MutexLock(&mu)" rather than
02133     // "<MutexLock l(&mu)". Hence the typeid trick below.
02134     class GTestMutexLock {
02135         public:
02136             explicit GTestMutexLock(MutexBase* mutex)
02137                 : mutex_(mutex) { mutex_->Lock(); }
02138
02139             ~GTestMutexLock() { mutex_->Unlock(); }
02140
02141         private:
02142             MutexBase* const mutex_;
02143
02144             GTEST_DISALLOW_COPY_AND_ASSIGN_(GTestMutexLock);
02145     };
02146
02147     typedef GTestMutexLock MutexLock;
02148
02149     // Helpers for ThreadLocal.
02150
02151     // pthread_key_create() requires DeleteThreadLocalValue() to have
02152     // C-linkage. Therefore it cannot be templatized to access
02153     // ThreadLocal<T>. Hence the need for class
02154     // ThreadLocalValueHolderBase.
02155     class ThreadLocalValueHolderBase {
02156         public:
02157             virtual ~ThreadLocalValueHolderBase() {}
02158     };
02159
02160     // Called by pthread to delete thread-local data stored by
02161     // pthread_setspecific().
02162     extern "C" inline void DeleteThreadLocalValue(void* value_holder) {
02163         delete static_cast<ThreadLocalValueHolderBase*>(value_holder);
02164     }
02165
02166     // Implements thread-local storage on pthreads-based systems.
02167     template <typename T>
02168     class GTEST_API_ ThreadLocal {
02169         public:
02170             ThreadLocal()
02171                 : key_(CreateKey()), default_factory_(new DefaultValueHolderFactory()) {}
02172             explicit ThreadLocal(const T& value)
02173                 : key_(CreateKey()),
02174                     default_factory_(new InstanceValueHolderFactory(value)) {}
02175
02176             ~ThreadLocal() {
02177                 // Destroys the managed object for the current thread, if any.
02178                 DeleteThreadLocalValue(pthread_getspecific(key_));
02179
02180                 // Releases resources associated with the key. This will *not*
02181                 // delete managed objects for other threads.
02182                 GTEST_CHECK_POSIX_SUCCESS_(pthread_key_delete(key_));
02183             }
02184
02185             T* pointer() { return GetOrCreateValue(); }
02186             const T* pointer() const { return GetOrCreateValue(); }
02187             const T& get() const { return *pointer(); }
02188             void set(const T& value) { *pointer() = value; }
02189
02190         private:
02191             // Holds a value of type T.
02192             class ValueHolder : public ThreadLocalValueHolderBase {
02193                 public:
02194                     ValueHolder() : value_() {}
02195                     explicit ValueHolder(const T& value) : value_(value) {}
02196
02197                     T* pointer() { return &value_; }
02198
02199                 private:
02200                     T value_;
02201                     GTEST_DISALLOW_COPY_AND_ASSIGN_(ValueHolder);
02202             };
02203
02204             static pthread_key_t CreateKey() {
02205                 pthread_key_t key;
02206                 // When a thread exits, DeleteThreadLocalValue() will be called on
02207                 // the object managed for that thread.
02208                 GTEST_CHECK_POSIX_SUCCESS_(
02209                     pthread_key_create(&key, &DeleteThreadLocalValue));
02210             return key;
```

```

02211 }
02212
02213 T* GetOrCreateValue() const {
02214     ThreadLocalValueHolderBase* const holder =
02215         static_cast<ThreadLocalValueHolderBase*>(pthread_getspecific(key_));
02216     if (holder != NULL) {
02217         return CheckedDowncastToActualType<ValueHolder>(holder)->pointer();
02218     }
02219
02220     ValueHolder* const new_holder = default_factory_->MakeNewHolder();
02221     ThreadLocalValueHolderBase* const holder_base = new_holder;
02222     GTEST_CHECK_POSIX_SUCCESS_(pthread_setspecific(key_, holder_base));
02223     return new_holder->pointer();
02224 }
02225
02226 class ValueHolderFactory {
02227 public:
02228     ValueHolderFactory() {}
02229     virtual ~ValueHolderFactory() {}
02230     virtual ValueHolder* MakeNewHolder() const = 0;
02231
02232 private:
02233     GTEST_DISALLOW_COPY_AND_ASSIGN_(ValueHolderFactory);
02234 };
02235
02236 class DefaultValueHolderFactory : public ValueHolderFactory {
02237 public:
02238     DefaultValueHolderFactory() {}
02239     virtual ValueHolder* MakeNewHolder() const { return new ValueHolder(); }
02240
02241 private:
02242     GTEST_DISALLOW_COPY_AND_ASSIGN_(DefaultValueHolderFactory);
02243 };
02244
02245 class InstanceValueHolderFactory : public ValueHolderFactory {
02246 public:
02247     explicit InstanceValueHolderFactory(const T& value) : value_(value) {}
02248     virtual ValueHolder* MakeNewHolder() const {
02249         return new ValueHolder(value_);
02250     }
02251
02252 private:
02253     const T value_; // The value for each thread.
02254
02255     GTEST_DISALLOW_COPY_AND_ASSIGN_(InstanceValueHolderFactory);
02256 };
02257
02258 // A key pthreads uses for looking up per-thread values.
02259 const pthread_key_t key_;
02260 scoped_ptr<ValueHolderFactory> default_factory_;
02261
02262 GTEST_DISALLOW_COPY_AND_ASSIGN_(ThreadLocal);
02263 };
02264
02265 #endif // GTEST_HAS_MUTEX_AND_THREAD_LOCAL_
02266
02267 #else // GTEST_IS_THREADSAFE
02268
02269 // A dummy implementation of synchronization primitives (mutex, lock,
02270 // and thread-local variable). Necessary for compiling Google Test where
02271 // mutex is not supported - using Google Test in multiple threads is not
02272 // supported on such platforms.
02273
02274 class Mutex {
02275 public:
02276     Mutex() {}
02277     void Lock() {}
02278     void Unlock() {}
02279     void AssertHeld() const {}
02280 };
02281
02282 # define GTEST_DECLARE_STATIC_MUTEX_(mutex) \
02283     extern ::testing::internal::Mutex mutex
02284
02285 # define GTEST_DEFINE_STATIC_MUTEX_(mutex) ::testing::internal::Mutex mutex
02286
02287 // We cannot name this class MutexLock because the ctor declaration would
02288 // conflict with a macro named MutexLock, which is defined on some
02289 // platforms. That macro is used as a defensive measure to prevent against
02290 // inadvertent misuses of MutexLock like "MutexLock(&mu)" rather than
02291 // "MutexLock l(&mu)". Hence the typedef trick below.
02292 class GTestMutexLock {
02293 public:
02294     explicit GTestMutexLock(Mutex*) {} // NOLINT
02295 };
02296
02297 typedef GTestMutexLock MutexLock;

```

```
02298 template <typename T>
02299 class GTEST_API_ ThreadLocal {
02300 public:
02301     ThreadLocal() : value_() {}
02302     explicit ThreadLocal(const T& value) : value_(value) {}
02303     T* pointer() { return &value_; }
02304     const T* pointer() const { return &value_; }
02305     const T& get() const { return value_; }
02306     void set(const T& value) { value_ = value; }
02307 private:
02308     T value_;
02309 };
02310 #endif // GTEST_IS_THREADSsafe
02311
02312 // Returns the number of threads running in the process, or 0 to indicate that
02313 // we cannot detect it.
02314 #define GTEST_API_ size_t GetThreadCount();
02315
02316 // Passing non-POD classes through ellipsis (...) crashes the ARM
02317 // compiler and generates a warning in Sun Studio before 12u4. The Nokia Symbian
02318 // and the IBM XL C/C++ compiler try to instantiate a copy constructor
02319 // for objects passed through ellipsis (...), failing for uncopyable
02320 // objects. We define this to ensure that only POD is passed through
02321 // ellipsis on these systems.
02322 #if defined(__SYMBIAN32__) || defined(__IBMCPP__)
02323     (defined(__SUNPRO_CC) && __SUNPRO_CC < 0x5130)
02324 // We lose support for NULL detection where the compiler doesn't like
02325 // passing non-POD classes through ellipsis (...).
02326 #define GTEST_ELLIPSIS_NEEDS_POD_ 1
02327
02328 #else
02329 #define GTEST_CAN_COMPARE_NULL 1
02330 #endif
02331
02332 // The Nokia Symbian and IBM XL C/C++ compilers cannot decide between
02333 // const T& and const T* in a function template. These compilers
02334 // _can_ decide between class template specializations for T and T*,
02335 // so a tr1::type_traits-like is_pointer works.
02336 #if defined(__SYMBIAN32__) || defined(__IBMCPP__)
02337 #define GTEST_NEEDS_IS_POINTER_ 1
02338 #endif
02339
02340 template <bool bool_value>
02341 struct bool_constant {
02342     typedef bool_constant<bool_value> type;
02343     static const bool value = bool_value;
02344 };
02345
02346 template <bool bool_value> const bool bool_constant<bool_value>::value;
02347
02348 typedef bool_constant<false> false_type;
02349 typedef bool_constant<true> true_type;
02350
02351 template <typename T, typename U>
02352 struct is_same : public false_type {};
02353
02354 template <typename T>
02355 struct is_same<T, T> : public true_type {};
02356
02357
02358 template <typename T>
02359 struct is_pointer : public false_type {};
02360
02361 template <typename T>
02362 struct is_pointer<T*> : public true_type {};
02363
02364 template <typename Iterator>
02365 struct IteratorTraits {
02366     typedef typename Iterator::value_type value_type;
02367 };
02368
02369
02370 template <typename T>
02371 struct IteratorTraits<T*> {
02372     typedef T value_type;
02373 };
02374
02375 template <typename T>
02376 struct IteratorTraits<const T*> {
02377     typedef T value_type;
02378 };
02379
02380 #if GTEST_OS_WINDOWS
02381 #define GTEST_PATH_SEP_ "\\"
02382 #define GTEST_HAS_ALT_PATH_SEP_ 1
02383 // The biggest signed integer type the compiler supports.
02384 typedef __int64 BiggestInt;
```

```

02385 #else
02386 # define GTEST_PATH_SEP_ "/"
02387 # define GTEST_HAS_ALT_PATH_SEP_ 0
02388 typedef long long BiggestInt; // NOLINT
02389 endif // GTEST_OS_WINDOWS
02390
02391 // Utilities for char.
02392
02393 // isspace(int ch) and friends accept an unsigned char or EOF. char
02394 // may be signed, depending on the compiler (or compiler flags).
02395 // Therefore we need to cast a char to unsigned char before calling
02396 // isspace(), etc.
02397
02398 inline bool IsAlpha(char ch) {
02399     return isalpha(static_cast<unsigned char>(ch)) != 0;
02400 }
02401 inline bool IsAlNum(char ch) {
02402     return isalnum(static_cast<unsigned char>(ch)) != 0;
02403 }
02404 inline bool IsDigit(char ch) {
02405     return isdigit(static_cast<unsigned char>(ch)) != 0;
02406 }
02407 inline bool IsLower(char ch) {
02408     return islower(static_cast<unsigned char>(ch)) != 0;
02409 }
02410 inline bool IsSpace(char ch) {
02411     return isspace(static_cast<unsigned char>(ch)) != 0;
02412 }
02413 inline bool IsUpper(char ch) {
02414     return isupper(static_cast<unsigned char>(ch)) != 0;
02415 }
02416 inline bool IsXDigit(char ch) {
02417     return isxdigit(static_cast<unsigned char>(ch)) != 0;
02418 }
02419 inline bool IsXDigit(wchar_t ch) {
02420     const unsigned char low_byte = static_cast<unsigned char>(ch);
02421     return ch == low_byte && isxdigit(low_byte) != 0;
02422 }
02423
02424 inline char ToLower(char ch) {
02425     return static_cast<char>(tolower(static_cast<unsigned char>(ch)));
02426 }
02427 inline char ToUpper(char ch) {
02428     return static_cast<char>(toupper(static_cast<unsigned char>(ch)));
02429 }
02430
02431 inline std::string StripTrailingSpaces(std::string str) {
02432     std::string::iterator it = str.end();
02433     while (it != str.begin() && IsSpace(*--it))
02434         it = str.erase(it);
02435     return str;
02436 }
02437
02438 // The testing::internal::posix namespace holds wrappers for common
02439 // POSIX functions. These wrappers hide the differences between
02440 // Windows/MSVC and POSIX systems. Since some compilers define these
02441 // standard functions as macros, the wrapper cannot have the same name
02442 // as the wrapped function.
02443
02444 namespace posix {
02445
02446 // Functions with a different name on Windows.
02447
02448 #if GTEST_OS_WINDOWS
02449
02450 typedef struct _stat StatStruct;
02451
02452 # ifdef __BORLANDC__
02453 inline int IsATTY(int fd) { return isatty(fd); }
02454 inline int StrCaseCmp(const char* s1, const char* s2) {
02455     return stricmp(s1, s2);
02456 }
02457 inline char* StrDup(const char* src) { return strdup(src); }
02458 # else // !__BORLANDC__
02459 # if GTEST_OS_WINDOWS_MOBILE
02460 inline int IsATTY(int /* fd */) { return 0; }
02461 # else
02462 inline int IsATTY(int fd) { return _isatty(fd); }
02463 # endif // GTEST_OS_WINDOWS_MOBILE
02464 inline int StrCaseCmp(const char* s1, const char* s2) {
02465     return _stricmp(s1, s2);
02466 }
02467 inline char* StrDup(const char* src) { return _strdup(src); }
02468 # endif // __BORLANDC__
02469
02470 # if GTEST_OS_WINDOWS_MOBILE
02471 inline int FileNo(FILE* file) { return reinterpret_cast<int>(_fileno(file)); }

```

```
02472 // Stat(), RmDir(), and IsDir() are not needed on Windows CE at this
02473 // time and thus not defined there.
02474 #else
02475 inline int FileNo(FILE* file) { return _fileno(file); }
02476 inline int Stat(const char* path, StatStruct* buf) { return _stat(path, buf); }
02477 inline int RmDir(const char* dir) { return _rmdir(dir); }
02478 inline bool IsDir(const StatStruct& st) {
02479     return (_S_IFDIR & st.st_mode) != 0;
02480 }
02481 #endif // GTEST_OS_WINDOWS_MOBILE
02482
02483 #else
02484
02485 typedef struct stat StatStruct;
02486
02487 inline int FileNo(FILE* file) { return fileno(file); }
02488 inline int Isatty(int fd) { return isatty(fd); }
02489 inline int Stat(const char* path, StatStruct* buf) { return stat(path, buf); }
02490 inline int StrCaseCmp(const char* s1, const char* s2) {
02491     return strcasecmp(s1, s2);
02492 }
02493 inline char* StrDup(const char* src) { return strdup(src); }
02494 inline int Rmdir(const char* dir) { return rmdir(dir); }
02495 inline bool IsDir(const StatStruct& st) { return S_ISDIR(st.st_mode); }
02496
02497 #endif // GTEST_OS_WINDOWS
02498
02499 // Functions deprecated by MSVC 8.0.
02500
02501 GTEST_DISABLE_MSC_DEPRECATED_PUSH_()
02502
02503 inline const char* StrNcpy(char* dest, const char* src, size_t n) {
02504     return strncpy(dest, src, n);
02505 }
02506
02507 // ChDir(), FReopen(), FDOpen(), Read(), Write(), Close(), and
02508 // StrError() aren't needed on Windows CE at this time and thus not
02509 // defined there.
02510
02511 #if !GTEST_OS_WINDOWS_MOBILE && !GTEST_OS_WINDOWS_PHONE && !GTEST_OS_WINDOWS_RT
02512 inline int ChDir(const char* dir) { return chdir(dir); }
02513 #endif
02514 inline FILE* FOpen(const char* path, const char* mode) {
02515     return fopen(path, mode);
02516 }
02517 #if !GTEST_OS_WINDOWS_MOBILE
02518 inline FILE *FReopen(const char* path, const char* mode, FILE* stream) {
02519     return freopen(path, mode, stream);
02520 }
02521 inline FILE* FDOpen(int fd, const char* mode) { return fdopen(fd, mode); }
02522 #endif
02523 inline int FClose(FILE* fp) { return fclose(fp); }
02524 #if !GTEST_OS_WINDOWS_MOBILE
02525 inline int Read(int fd, void* buf, unsigned int count) {
02526     return static_cast<int>(read(fd, buf, count));
02527 }
02528 inline int Write(int fd, const void* buf, unsigned int count) {
02529     return static_cast<int>(write(fd, buf, count));
02530 }
02531 inline int Close(int fd) { return close(fd); }
02532 inline const char* StrError(int errnum) { return strerror(errnum); }
02533 #endif
02534 inline const char* GetEnv(const char* name) {
02535 #if GTEST_OS_WINDOWS_MOBILE || GTEST_OS_WINDOWS_PHONE || GTEST_OS_WINDOWS_RT
02536     // We are on Windows CE, which has no environment variables.
02537     static_cast<void>(name); // To prevent 'unused argument' warning.
02538     return NULL;
02539 #elif defined(__BORLANDC__) || defined(__SunOS_5_8) || defined(__SunOS_5_9)
02540     // Environment variables which we programmatically clear will be set to the
02541     // empty string rather than unset (NULL). Handle that case.
02542     const char* const env = getenv(name);
02543     return (env != NULL && env[0] != '\0') ? env : NULL;
02544 #else
02545     return getenv(name);
02546 #endif
02547 }
02548
02549 GTEST_DISABLE_MSC_DEPRECATED_POP_()
02550
02551 #if GTEST_OS_WINDOWS_MOBILE
02552 // Windows CE has no C library. The abort() function is used in
02553 // several places in Google Test. This implementation provides a reasonable
02554 // imitation of standard behaviour.
02555 void Abort();
02556 #else
02557 inline void Abort() { abort(); }
02558 #endif // GTEST_OS_WINDOWS_MOBILE
```

```

02559 } // namespace posix
02560
02561
02562 // MSVC "deprecates" snprintf and issues warnings wherever it is used. In
02563 // order to avoid these warnings, we need to use _snprintf or _snprintf_s on
02564 // MSVC-based platforms. We map the GTEST_SNPRINTF_ macro to the appropriate
02565 // function in order to achieve that. We use macro definition here because
02566 // snprintf is a variadic function.
02567 #if _MSC_VER >= 1400 && !GTEST_OS_WINDOWS_MOBILE
02568 // MSVC 2005 and above support variadic macros.
02569 # define GTEST_SNPRINTF_(buffer, size, format, ...) \
02570     _snprintf_s(buffer, size, size, format, __VA_ARGS__)
02571 #elif defined(_MSC_VER)
02572 // Windows CE does not define _snprintf_s and MSVC prior to 2005 doesn't
02573 // complain about _snprintf.
02574 # define GTEST_SNPRINTF_ _snprintf
02575 #else
02576 # define GTEST_SNPRINTF_ snprintf
02577 #endif
02578
02579 // The maximum number a BiggestInt can represent. This definition
02580 // works no matter BiggestInt is represented in one's complement or
02581 // two's complement.
02582 //
02583 // We cannot rely on numeric_limits in STL, as __int64 and long long
02584 // are not part of standard C++ and numeric_limits doesn't need to be
02585 // defined for them.
02586 const BiggestInt kMaxBiggestInt =
02587     ~(static_cast<BiggestInt>(1) << (8*sizeof(BiggestInt) - 1));
02588
02589 // This template class serves as a compile-time function from size to
02590 // type. It maps a size in bytes to a primitive type with that
02591 // size. e.g.
02592 //
02593 //    TypeWithSize<4>::UInt
02594 //
02595 // is typedef-ed to be unsigned int (unsigned integer made up of 4
02596 // bytes).
02597 //
02598 // Such functionality should belong to STL, but I cannot find it
02599 // there.
02600 //
02601 // Google Test uses this class in the implementation of floating-point
02602 // comparison.
02603 //
02604 // For now it only handles UInt (unsigned int) as that's all Google Test
02605 // needs. Other types can be easily added in the future if need
02606 // arises.
02607 template <size_t size>
02608 class TypeWithSize {
02609 public:
02610     // This prevents the user from using TypeWithSize<N> with incorrect
02611     // values of N.
02612     typedef void UInt;
02613 };
02614
02615 // The specialization for size 4.
02616 template <>
02617 class TypeWithSize<4> {
02618 public:
02619     // unsigned int has size 4 in both gcc and MSVC.
02620     //
02621     // As base/basicitypes.h doesn't compile on Windows, we cannot use
02622     // uint32, uint64, and etc here.
02623     typedef int Int;
02624     typedef unsigned int UInt;
02625 };
02626
02627 // The specialization for size 8.
02628 template <>
02629 class TypeWithSize<8> {
02630 public:
02631 #if GTEST_OS_WINDOWS
02632     typedef __int64 Int;
02633     typedef unsigned __int64 UInt;
02634 #else
02635     typedef long long Int; // NOLINT
02636     typedef unsigned long long UInt; // NOLINT
02637 #endif // GTEST_OS_WINDOWS
02638 };
02639
02640 // Integer types of known sizes.
02641 typedef TypeWithSize<4>::Int Int32;
02642 typedef TypeWithSize<4>::UInt UInt32;
02643 typedef TypeWithSize<8>::Int Int64;
02644 typedef TypeWithSize<8>::UInt UInt64;
02645 typedef TypeWithSize<8>::Int TimeInMillis; // Represents time in milliseconds.

```

```
02646
02647 // Utilities for command line flags and environment variables.
02648
02649 // Macro for referencing flags.
02650 #if !defined(GTEST_FLAG)
02651 # define GTEST_FLAG(name) FLAGS_gtest_##name
02652 #endif // !defined(GTEST_FLAG)
02653
02654 #if !defined(GTEST_USE_OWN_FLAGFILE_FLAG_)
02655 # define GTEST_USE_OWN_FLAGFILE_FLAG_ 1
02656 #endif // !defined(GTEST_USE_OWN_FLAGFILE_FLAG_)
02657
02658 #if !defined(GTEST_DECLARE_bool_)
02659 # define GTEST_FLAG_SAVER_ ::testing::internal::GTestFlagSaver
02660
02661 // Macros for declaring flags.
02662 # define GTEST_DECLARE_bool_(name) GTEST_API_ extern bool GTEST_FLAG(name)
02663 # define GTEST_DECLARE_int32_(name) \
02664     GTEST_API_ extern ::testing::internal::Int32 GTEST_FLAG(name)
02665 # define GTEST_DECLARE_string_(name) \
02666     GTEST_API_ extern ::std::string GTEST_FLAG(name)
02667
02668 // Macros for defining flags.
02669 # define GTEST_DEFINE_bool_(name, default_val, doc) \
02670     GTEST_API_ bool GTEST_FLAG(name) = (default_val)
02671 # define GTEST_DEFINE_int32_(name, default_val, doc) \
02672     GTEST_API_ ::testing::internal::Int32 GTEST_FLAG(name) = (default_val)
02673 # define GTEST_DEFINE_string_(name, default_val, doc) \
02674     GTEST_API_ ::std::string GTEST_FLAG(name) = (default_val)
02675
02676 #endif // !defined(GTEST_DECLARE_bool_)
02677
02678 // Thread annotations
02679 #if !defined(GTEST_EXCLUSIVE_LOCK_REQUIRED_)
02680 # define GTEST_EXCLUSIVE_LOCK_REQUIRED_(locks)
02681 # define GTEST_LOCK_EXCLUDED_(locks)
02682 #endif // !defined(GTEST_EXCLUSIVE_LOCK_REQUIRED_)
02683
02684 // Parses 'str' for a 32-bit signed integer. If successful, writes the result
02685 // to *value and returns true; otherwise leaves *value unchanged and returns
02686 // false.
02687 // FIXME: Find a better way to refactor flag and environment parsing
02688 // out of both gtest-port.cc and gtest.cc to avoid exporting this utility
02689 // function.
02690 bool ParseInt32(const Message& src_text, const char* str, Int32* value);
02691
02692 // Parses a bool/Int32/string from the environment variable
02693 // corresponding to the given Google Test flag.
02694 bool BoolFromGTestEnv(const char* flag, bool default_val);
02695 GTEST_API_ Int32 Int32FromGTestEnv(const char* flag, Int32 default_val);
02696 std::string OutputFlagAlsoCheckEnvVar();
02697 const char* StringFromGTestEnv(const char* flag, const char* default_val);
02698
02699 } // namespace internal
02700 } // namespace testing
02701
02702 #endif // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PORT_H_
```

9.32 Dokumentacja pliku

packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/custom/README.md

9.33 Dokumentacja pliku

packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-death-test-internal.h

```
#include "gtest/internal/gtest-internal.h"
#include <stdio.h>
```

Przestrzenie nazw

- namespace `testing`
- namespace `testing::internal`

Funkcje

- `testing::internal::GTEST_DECLARE_string_ (internal_run_death_test)`

Zmienne

- const char `testing::internal::kDeathTestStyleFlag [] = "death_test_style"`
- const char `testing::internal::kDeathTestUseFork [] = "death_test_use_fork"`
- const char `testing::internal::kInternalRunDeathTestFlag [] = "internal_run_death_test"`

9.34 gtest-death-test-internal.h

[Idź do dokumentacji tego pliku.](#)

```

00001 // Copyright 2005, Google Inc.
00002 // All rights reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //      * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //      * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 // The Google C++ Testing and Mocking Framework (Google Test)
00031 //
00032 // This header file defines internal utilities needed for implementing
00033 // death tests. They are subject to change without notice.
00034 // GOOGLETEST_CM0001 DO NOT DELETE
00035
00036 #ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_DEATH_TEST_INTERNAL_H_
00037 #define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_DEATH_TEST_INTERNAL_H_
00038
00039 #include "gtest/internal/gtest-internal.h"
00040
00041 #include <stdio.h>
00042
00043 namespace testing {
00044 namespace internal {
00045
00046 GTEST_DECLARE_string_(internal_run_death_test);
00047
00048 // Names of the flags (needed for parsing Google Test flags).
00049 const char kDeathTestStyleFlag[] = "death_test_style";
00050 const char kDeathTestUseFork[] = "death_test_use_fork";
00051 const char kInternalRunDeathTestFlag[] = "internal_run_death_test";
00052

```

```
00053 #if GTEST_HAS_DEATH_TEST
00054
00055 GTEST_DISABLE_MSC_WARNINGS_PUSH_(4251 \
00056 /* class A needs to have dll-interface to be used by clients of class B */
00057
00058 // DeathTest is a class that hides much of the complexity of the
00059 // GTEST_DEATH_TEST_ macro. It is abstract; its static Create method
00060 // returns a concrete class that depends on the prevailing death test
00061 // style, as defined by the --gtest_death_test_style and/or
00062 // --gtest_internal_run_death_test flags.
00063
00064 // In describing the results of death tests, these terms are used with
00065 // the corresponding definitions:
00066 //
00067 // exit status: The integer exit information in the format specified
00068 // by wait(2)
00069 // exit code: The integer code passed to exit(3), _exit(2), or
00070 // returned from main()
00071 class GTEST_API_ DeathTest {
00072 public:
00073     // Create returns false if there was an error determining the
00074     // appropriate action to take for the current death test; for example,
00075     // if the gtest_death_test_style flag is set to an invalid value.
00076     // The LastMessage method will return a more detailed message in that
00077     // case. Otherwise, the DeathTest pointer pointed to by the "test"
00078     // argument is set. If the death test should be skipped, the pointer
00079     // is set to NULL; otherwise, it is set to the address of a new concrete
00080     // DeathTest object that controls the execution of the current test.
00081     static bool Create(const char* statement, const RE* regex,
00082                         const char* file, int line, DeathTest** test);
00083     DeathTest();
00084     virtual ~DeathTest() { }
00085
00086     // A helper class that aborts a death test when it's deleted.
00087     class ReturnSentinel {
00088         public:
00089             explicit ReturnSentinel(DeathTest* test) : test_(test) { }
00090             ~ReturnSentinel() { test_->Abort(TEST_ENCONTRIED_RETURN_STATEMENT); }
00091         private:
00092             DeathTest* const test_;
00093             GTEST_DISALLOW_COPY_AND_ASSIGN_(ReturnSentinel);
00094     } GTEST_ATTRIBUTE_UNUSED_;
00095
00096     // An enumeration of possible roles that may be taken when a death
00097     // test is encountered. EXECUTE means that the death test logic should
00098     // be executed immediately. OVERSEE means that the program should prepare
00099     // the appropriate environment for a child process to execute the death
00100     // test, then wait for it to complete.
00101     enum TestRole { OVERSEE_TEST, EXECUTE_TEST };
00102
00103     // An enumeration of the three reasons that a test might be aborted.
00104     enum AbortReason {
00105         TEST_ENCONTRIED_RETURN_STATEMENT,
00106         TEST_THROWED_EXCEPTION,
00107         TEST_DID_NOT_DIE
00108     };
00109
00110     // Assumes one of the above roles.
00111     virtual TestRole AssumeRole() = 0;
00112
00113     // Waits for the death test to finish and returns its status.
00114     virtual int Wait() = 0;
00115
00116     // Returns true if the death test passed; that is, the test process
00117     // exited during the test, its exit status matches a user-supplied
00118     // predicate, and its stderr output matches a user-supplied regular
00119     // expression.
00120     // The user-supplied predicate may be a macro expression rather
00121     // than a function pointer or functor, or else Wait and Passed could
00122     // be combined.
00123     virtual bool Passed(bool exit_status_ok) = 0;
00124
00125     // Signals that the death test did not die as expected.
00126     virtual void Abort(AbortReason reason) = 0;
00127
00128     // Returns a human-readable outcome message regarding the outcome of
00129     // the last death test.
00130     static const char* LastMessage();
00131
00132     static void set_last_death_test_message(const std::string& message);
00133
00134     private:
00135     // A string containing a description of the outcome of the last death test.
00136     static std::string last_death_test_message_;
00137
00138     GTEST_DISALLOW_COPY_AND_ASSIGN_(DeathTest);
00139 }
```

```

00140
00141 GTEST_DISABLE_MSC_WARNINGS_POP_() // 4251
00142
00143 // Factory interface for death tests. May be mocked out for testing.
00144 class DeathTestFactory {
00145 public:
00146 virtual ~DeathTestFactory() { }
00147 virtual bool Create(const char* statement, const RE* regex,
00148 const char* file, int line, DeathTest** test) = 0;
00149 };
00150
00151 // A concrete DeathTestFactory implementation for normal use.
00152 class DefaultDeathTestFactory : public DeathTestFactory {
00153 public:
00154 virtual bool Create(const char* statement, const RE* regex,
00155 const char* file, int line, DeathTest** test);
00156 };
00157
00158 // Returns true if exit_status describes a process that was terminated
00159 // by a signal, or exited normally with a nonzero exit code.
00160 GTEST_API_ bool ExitedUnsuccessfully(int exit_status);
00161
00162 // Traps C++ exceptions escaping statement and reports them as test
00163 // failures. Note that trapping SEH exceptions is not implemented here.
00164 #if GTEST_HAS_EXCEPTIONS
00165 #define GTEST_EXECUTE_DEATH_TEST_STATEMENT_(statement, death_test) \
00166 try { \
00167     GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement); \
00168 } catch (const ::std::exception& gtest_exception) { \
00169     fprintf( \
00170         stderr, \
00171         "\n%s: Caught std::exception-derived exception escaping the " \
00172         "death test statement. Exception message: %s\n", \
00173         ::testing::internal::FormatFileLocation(__FILE__, __LINE__).c_str(), \
00174         gtest_exception.what()); \
00175     fflush(stderr); \
00176     death_test->Abort(::testing::internal::DeathTest::TEST_THROWN_EXCEPTION); \
00177 } catch (...) { \
00178     death_test->Abort(::testing::internal::DeathTest::TEST_THROWN_EXCEPTION); \
00179 }
00180
00181 #else
00182 #define GTEST_EXECUTE_DEATH_TEST_STATEMENT_(statement, death_test) \
00183     GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement)
00184
00185 #endif
00186
00187 // This macro is for implementing ASSERT_DEATH*, EXPECT_DEATH*,
00188 // ASSERT_EXIT*, and EXPECT_EXIT*.
00189 #define GTEST_DEATH_TEST_(statement, predicate, regex, fail) \
00190     GTEST_AMBIGUOUS_ELSE_BLOCKER_ \
00191     if (::testing::internal::AlwaysTrue()) { \
00192         const ::testing::internal::RE& gtest_regex = (regex); \
00193         ::testing::internal::DeathTest* gtest_dt; \
00194         if (!::testing::internal::DeathTest::Create(#statement, &gtest_regex, \
00195             __FILE__, __LINE__, &gtest_dt)) { \
00196             goto GTEST_CONCAT_TOKEN_(gtest_label_, __LINE__); \
00197         } \
00198         if (gtest_dt != NULL) { \
00199             ::testing::internal::scoped_ptr<::testing::internal::DeathTest> \
00200                 gtest_dt_ptr(gtest_dt); \
00201             switch (gtest_dt->AssumeRole()) { \
00202                 case ::testing::internal::DeathTest::OVERSEE_TEST: \
00203                     if (!gtest_dt->Passed(predicate(gtest_dt->Wait()))) { \
00204                         goto GTEST_CONCAT_TOKEN_(gtest_label_, __LINE__); \
00205                     } \
00206                     break; \
00207                 case ::testing::internal::DeathTest::EXECUTE_TEST: { \
00208                     ::testing::internal::DeathTest::ReturnSentinel \
00209                         gtest_sentinel(gtest_dt); \
00210                     GTEST_EXECUTE_DEATH_TEST_STATEMENT_(statement, gtest_dt); \
00211                     gtest_dt->Abort(::testing::internal::DeathTest::TEST_DID_NOT_DIE); \
00212                     break; \
00213                 } \
00214                 default: \
00215                     break; \
00216             } \
00217         } \
00218     } else { \
00219         GTEST_CONCAT_TOKEN_(gtest_label_, __LINE__): \
00220             fail(::testing::internal::DeathTest::LastMessage())
00221 // The symbol "fail" here expands to something into which a message
00222 // can be streamed.
00223
00224 // This macro is for implementing ASSERT/EXPECT_DEBUG_DEATH when compiled in
00225 // NDEBUG mode. In this case we need the statements to be executed and the macro
00226 // must accept a streamed message even though the message is never printed.

```

```

00227 // The regex object is not evaluated, but it is used to prevent "unused"
00228 // warnings and to avoid an expression that doesn't compile in debug mode.
00229 #define GTEST_EXECUTE_STATEMENT_(statement, regex)
00230     GTEST_AMBIGUOUS_ELSE_BLOCKER_
00231     if (::testing::internal::AlwaysTrue()) {
00232         GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement);
00233     } else if (!::testing::internal::AlwaysTrue()) {
00234         const ::testing::internal::RE& gtest_regex = (regex);
00235         static_cast<void>(gtest_regex);
00236     } else
00237         ::testing::Message()
00238
00239 // A class representing the parsed contents of the
00240 // --gtest_internal_run_death_test flag, as it existed when
00241 // RUN_ALL_TESTS was called.
00242 class InternalRunDeathTestFlag {
00243 public:
00244     InternalRunDeathTestFlag(const std::string& a_file,
00245                             int a_line,
00246                             int an_index,
00247                             int a_write_fd)
00248     : file_(a_file), line_(a_line), index_(an_index),
00249       write_fd_(a_write_fd) {}
00250
00251 ~InternalRunDeathTestFlag() {
00252     if (write_fd_ >= 0)
00253         posix::Close(write_fd_);
00254 }
00255
00256 const std::string& file() const { return file_; }
00257 int line() const { return line_; }
00258 int index() const { return index_; }
00259 int write_fd() const { return write_fd_; }
00260
00261 private:
00262     std::string file_;
00263     int line_;
00264     int index_;
00265     int write_fd_;
00266
00267     GTEST_DISALLOW_COPY_AND_ASSIGN_(InternalRunDeathTestFlag);
00268 };
00269
00270 // Returns a newly created InternalRunDeathTestFlag object with fields
00271 // initialized from the GTEST_FLAG(internal_run_death_test) flag if
00272 // the flag is specified; otherwise returns NULL.
00273 InternalRunDeathTestFlag* ParseInternalRunDeathTestFlag();
00274
00275 #endif // GTEST_HAS_DEATH_TEST
00276
00277 } // namespace internal
00278 } // namespace testing
00279
00280 #endif // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_DEATH_TEST_INTERNAL_H_

```

9.35 Dokumentacja pliku packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-filepath.h

```
#include "gtest/internal/gtest-string.h"
```

Funkcje

- [GTEST_DISABLE_MSC_WARNINGS_PUSH_](#) (4251) namespace **testing**

9.35.1 Dokumentacja funkcji

9.35.1.1 GTEST_DISABLE_MSC_WARNINGS_PUSH_()

```
GTEST_DISABLE_MSC_WARNINGS_PUSH_ (
    4251 )
```

9.36 gtest-filepath.h

[Idź do dokumentacji tego pliku.](#)

```

00001 // Copyright 2008, Google Inc.
00002 // All rights reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //      * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //      * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 // Google Test filepath utilities
00031 //
00032 // This header file declares classes and functions used internally by
00033 // Google Test. They are subject to change without notice.
00034 //
00035 // This file is #included in gtest/internal/gtest-internal.h.
00036 // Do not include this header file separately!
00037
00038 // GOOGLETEST_CM0001 DO NOT DELETE
00039
00040 #ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_FILEPATH_H_
00041 #define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_FILEPATH_H_
00042
00043 #include "gtest/internal/gtest-string.h"
00044
00045 GTEST_DISABLE_MSC_WARNINGS_PUSH_(4251 \
00046 /* class A needs to have dll-interface to be used by clients of class B */)
00047
00048 namespace testing {
00049 namespace internal {
00050
00051 // FilePath - a class for file and directory pathname manipulation which
00052 // handles platform-specific conventions (like the pathname separator).
00053 // Used for helper functions for naming files in a directory for xml output.
00054 // Except for Set methods, all methods are const or static, which provides an
00055 // "immutable value object" -- useful for peace of mind.
00056 // A FilePath with a value ending in a path separator ("like/this/") represents
00057 // a directory, otherwise it is assumed to represent a file. In either case,
00058 // it may or may not represent an actual file or directory in the file system.
00059 // Names are NOT checked for syntax correctness -- no checking for illegal
00060 // characters, malformed paths, etc.
00061
00062 class GTEST_API_ FilePath {
00063 public:
00064     FilePath() : pathname_("") { }
00065     FilePath(const FilePath& rhs) : pathname_(rhs.pathname_) { }
00066
00067     explicit FilePath(const std::string& pathname) : pathname_(pathname) {
00068         Normalize();
00069     }
00070
00071     FilePath& operator=(const FilePath& rhs) {
00072         Set(rhs);
00073         return *this;
00074     }
00075
00076     void Set(const FilePath& rhs) {
00077         pathname_ = rhs.pathname_;
00078     }
00079
00080     const std::string& string() const { return pathname_; }
00081     const char* c_str() const { return pathname_.c_str(); }
00082

```

```
00083 // Returns the current working directory, or "" if unsuccessful.
00084 static FilePath GetCurrentDir();
00085
00086 // Given directory = "dir", base_name = "test", number = 0,
00087 // extension = "xml", returns "dir/test.xml". If number is greater
00088 // than zero (e.g., 12), returns "dir/test_12.xml".
00089 // On Windows platform, uses \ as the separator rather than /.
00090 static FilePath MakeFileName(const FilePath& directory,
00091                         const FilePath& base_name,
00092                         int number,
00093                         const char* extension);
00094
00095 // Given directory = "dir", relative_path = "test.xml",
00096 // returns "dir/test.xml".
00097 // On Windows, uses \ as the separator rather than /.
00098 static FilePath ConcatPaths(const FilePath& directory,
00099                         const FilePath& relative_path);
00100
00101 // Returns a pathname for a file that does not currently exist. The pathname
00102 // will be directory/base_name.extension or
00103 // directory/base_name_<number>.extension if directory/base_name.extension
00104 // already exists. The number will be incremented until a pathname is found
00105 // that does not already exist.
00106 // Examples: 'dir/foo_test.xml' or 'dir/foo_test_1.xml'.
00107 // There could be a race condition if two or more processes are calling this
00108 // function at the same time -- they could both pick the same filename.
00109 static FilePath GenerateUniqueFileName(const FilePath& directory,
00110                         const FilePath& base_name,
00111                         const char* extension);
00112
00113 // Returns true iff the path is "".
00114 bool IsEmpty() const { return pathname_.empty(); }
00115
00116 // If input name has a trailing separator character, removes it and returns
00117 // the name, otherwise return the name string unmodified.
00118 // On Windows platform, uses \ as the separator, other platforms use /.
00119 FilePath RemoveTrailingPathSeparator() const;
00120
00121 // Returns a copy of the FilePath with the directory part removed.
00122 // Example: FilePath("path/to/file").RemoveDirectoryName() returns
00123 // FilePath("file"). If there is no directory part ("just_a_file"), it returns
00124 // the FilePath unmodified. If there is no file part ("just_a_dir/") it
00125 // returns an empty FilePath ("").
00126 // On Windows platform, '\' is the path separator, otherwise it is '/'.
00127 FilePath RemoveDirectoryName() const;
00128
00129 // RemoveFileName returns the directory path with the filename removed.
00130 // Example: FilePath("path/to/file").RemoveFileName() returns "path/to/".
00131 // If the FilePath is "a_file" or "/a_file", RemoveFileName returns
00132 // FilePath("./") or, on Windows, FilePath(".\\"). If the filepath does
00133 // not have a file, like "just/a/dir/", it returns the FilePath unmodified.
00134 // On Windows platform, '\' is the path separator, otherwise it is '/'.
00135 FilePath RemoveFileName() const;
00136
00137 // Returns a copy of the FilePath with the case-insensitive extension removed.
00138 // Example: FilePath("dir/file.exe").RemoveExtension("EXE") returns
00139 // FilePath("dir/file"). If a case-insensitive extension is not
00140 // found, returns a copy of the original FilePath.
00141 FilePath RemoveExtension(const char* extension) const;
00142
00143 // Creates directories so that path exists. Returns true if successful or if
00144 // the directories already exist; returns false if unable to create
00145 // directories for any reason. Will also return false if the FilePath does
00146 // not represent a directory (that is, it doesn't end with a path separator).
00147 bool CreateDirectoriesRecursively() const;
00148
00149 // Create the directory so that path exists. Returns true if successful or
00150 // if the directory already exists; returns false if unable to create the
00151 // directory for any reason, including if the parent directory does not
00152 // exist. Not named "CreateDirectory" because that's a macro on Windows.
00153 bool CreateFolder() const;
00154
00155 // Returns true if FilePath describes something in the file-system,
00156 // either a file, directory, or whatever, and that something exists.
00157 bool FileOrDirectoryExists() const;
00158
00159 // Returns true if pathname describes a directory in the file-system
00160 // that exists.
00161 bool DirectoryExists() const;
00162
00163 // Returns true if FilePath ends with a path separator, which indicates that
00164 // it is intended to represent a directory. Returns false otherwise.
00165 // This does NOT check that a directory (or file) actually exists.
00166 bool IsDirectory() const;
00167
00168 // Returns true if pathname describes a root directory. (Windows has one
00169 // root directory per disk drive.)
```

```

00170     bool IsRootDirectory() const;
00171
00172     // Returns true if pathname describes an absolute path.
00173     bool IsAbsolutePath() const;
00174
00175     private:
00176     // Replaces multiple consecutive separators with a single separator.
00177     // For example, "bar///foo" becomes "bar/foo". Does not eliminate other
00178     // redundancies that might be in a pathname involving ".." or ...
00179     //
00180     // A pathname with multiple consecutive separators may occur either through
00181     // user error or as a result of some scripts or APIs that generate a pathname
00182     // with a trailing separator. On other platforms the same API or script
00183     // may NOT generate a pathname with a trailing "/". Then elsewhere that
00184     // pathname may have another "/" and pathname components added to it,
00185     // without checking for the separator already being there.
00186     // The script language and operating system may allow paths like "foo//bar"
00187     // but some of the functions in FilePath will not handle that correctly. In
00188     // particular, RemoveTrailingPathSeparator() only removes one separator, and
00189     // it is called in CreateDirectoriesRecursively() assuming that it will change
00190     // a pathname from directory syntax (trailing separator) to filename syntax.
00191     //
00192     // On Windows this method also replaces the alternate path separator '\/' with
00193     // the primary path separator '\\', so that for example "bar\\\/\\foo" becomes
00194     // "bar\\foo".
00195
00196     void Normalize();
00197
00198     // Returns a pointer to the last occurrence of a valid path separator in
00199     // the FilePath. On Windows, for example, both '/' and '\' are valid path
00200     // separators. Returns NULL if no path separator was found.
00201     const char* FindLastPathSeparator() const;
00202
00203     std::string pathname_;
00204 }; // class FilePath
00205
00206 } // namespace internal
00207 } // namespace testing
00208
00209 GTEST_DISABLE_MSC_WARNINGS_POP_() // 4251
00210
00211 #endif // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_FILEPATH_H_

```

9.37 Dokumentacja pliku

packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-internal.h

```

#include "gtest/internal/gtest-port.h"
#include <ctype.h>
#include <float.h>
#include <string.h>
#include <iomanip>
#include <limits>
#include <map>
#include <set>
#include <string>
#include <vector>
#include "gtest/gtest-message.h"
#include "gtest/internal/gtest-filepath.h"
#include "gtest/internal/gtest-string.h"
#include "gtest/internal/gtest-type-util.h"

```

Komponenty

- class **testing::internal::FloatingPoint< RawType >**
- class **testing::internal::TypeldHelper< T >**

- class `testing::internal::TestFactoryBase`
- class `testing::internal::TestFactoryImpl< TestClass >`
- struct `testing::internal::CodeLocation`
- struct `testing::internal::ConstCharPtr`
- class `testing::internal::Random`
- struct `testing::internal::CompileAssertTypesEqual< T, T >`
- struct `testing::internal::RemoveReference< T >`
- struct `testing::internal::RemoveReference< T & >`
- struct `testing::internal::RemoveConst< T >`
- struct `testing::internal::RemoveConst< const T >`
- struct `testing::internal::RemoveConst< const T[N]>`
- class `testing::internal::ImplicitlyConvertible< From, To >`
- struct `testing::internal::IsAProtocolMessage< T >`
- struct `testing::internal::IsHashTable< T >`
- struct `testing::internal::VoidT< T >`
- struct `testing::internal::HasValueType< T, typename >`
- struct `testing::internal::HasValueType< T, VoidT< typename T::value_type > >`
- struct `testing::internal::IsRecursiveContainerImpl< C, false, HV >`
- struct `testing::internal::IsRecursiveContainerImpl< C, true, false >`
- struct `testing::internal::IsRecursiveContainerImpl< C, true, true >`
- struct `testing::internal::IsRecursiveContainer< C >`
- struct `testing::internal::EnableIf< true >`
- struct `testing::internal::RelationToSourceReference`
- struct `testing::internal::RelationToSourceCopy`
- class `testing::internal::NativeArray< Element >`

Przestrzenie nazw

- namespace `proto2`
- namespace `testing`
- namespace `testing::internal`
- namespace `testing::internal::edit_distance`

Definicje

- `#define GTEST_CONCAT_TOKEN_(foo, bar)`
- `#define GTEST_CONCAT_TOKEN_IMPL_(foo, bar)`
- `#define GTEST_STRINGIFY_(name)`
- `#define GTEST_IS_NULL_LITERAL_(x)`
- `#define GTEST_REMOVE_REFERENCE_(T)`
- `#define GTEST_REMOVE_CONST_(T)`
- `#define GTEST_REMOVE_REFERENCE_AND_CONST_(T)`
- `#define GTEST_MESSAGE_AT_(file, line, message, result_type)`
- `#define GTEST_MESSAGE_(message, result_type)`
- `#define GTEST_FATAL_FAILURE_(message)`
- `#define GTEST_NONFATAL_FAILURE_(message)`
- `#define GTEST_SUCCESS_(message)`
- `#define GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement)`
- `#define GTEST_TEST_THROW_(statement, expected_exception, fail)`
- `#define GTEST_TEST_NO_THROW_(statement, fail)`
- `#define GTEST_TEST_ANY_THROW_(statement, fail)`
- `#define GTEST_TEST_BOOLEAN_(expression, text, actual, expected, fail)`
- `#define GTEST_TEST_NO_FATAL_FAILURE_(statement, fail)`
- `#define GTEST_TEST_CLASS_NAME_(test_case_name, test_name)`
- `#define GTEST_TEST_(test_case_name, test_name, parent_class, parent_id)`

Definicje typów

- `typedef FloatingPoint< float > testing::internal::Float`
- `typedef FloatingPoint< double > testing::internal::Double`
- `typedef const void * testing::internal::Typeld`
- `typedef void(* testing::internal::SetUpTestCaseFunc) ()`
- `typedef void(* testing::internal::TearDownTestCaseFunc) ()`
- `typedef int testing::internal::IsContainer`
- `typedef char testing::internal::IsNotContainer`

Wyliczenia

- enum `testing::internal::edit_distance::EditType { testing::internal::edit_distance::kMatch , testing::internal::edit_distance::kAdd , testing::internal::edit_distance::kRemove , testing::internal::edit_distance::kReplace }`

Funkcje

- template<typename T>
 `::std::string testing::PrintToString (const T &value)`
- `char testing::internal::IsNullLiteralHelper (Secret *p)`
- `char(& testing::internal::IsNullLiteralHelper (...))[2]`
- `GTEST_API_ std::string testing::internal::AppendUserMessage (const std::string >est_msg, const Message &user_msg)`
- `GTEST_API_ std::vector< EditType > testing::internal::CalculateOptimalEdits (const std::vector< size_t > &left, const std::vector< size_t > &right)`
- `GTEST_API_ std::vector< EditType > testing::internal::CalculateOptimalEdits (const std::vector< std::string > &left, const std::vector< std::string > &right)`
- `GTEST_API_ std::string testing::internal::CreateUnifiedDiff (const std::vector< std::string > &left, const std::vector< std::string > &right, size_t context=2)`
- `GTEST_API_ std::string testing::internal::DiffStrings (const std::string &left, const std::string &right, size_t *total_line_count)`
- `GTEST_API_ AssertionResult testing::internal::EqFailure (const char *expected_expression, const char *actual_expression, const std::string &expected_value, const std::string &actual_value, bool ignoring_case)`
- `GTEST_API_ std::string testing::internal::GetBoolAssertionFailureMessage (const AssertionResult &assertion_result, const char *expression_text, const char *actual_predicate_value, const char *expected_predicate_value)`
- template<typename T>
 `Typeld testing::internal::GetTypeld ()`
- `GTEST_API_ Typeld testing::internal::GetTestTypeld ()`
- `GTEST_API_ TestInfo * testing::internal::MakeAndRegisterTestInfo (const char *test_case_name, const char *name, const char *type_param, const char *value_param, CodeLocation code_location, Typeld fixture_class_id, SetUpTestCaseFunc set_up_tc, TearDownTestCaseFunc tear_down_tc, TestFactoryBase *factory)`
- `GTEST_API_ bool testing::internal::SkipPrefix (const char *prefix, const char **ppstr)`
- `GTEST_API_ std::string testing::internal::GetCurrentOsStackTraceExceptTop (UnitTest *unit_test, int skip_count)`
- `GTEST_API_ bool testing::internal::AlwaysTrue ()`
- `bool testing::internal::AlwaysFalse ()`
- template<class C>
 `IsContainer testing::internal::IsContainerTest (int, typename C::iterator *=NULL, typename C::const_iterator *=NULL)`
- template<class C>
 `IsNotContainer testing::internal::IsContainerTest (long)`
- template<typename T, typename U>
 `bool testing::internal::ArrayEq (const T *lhs, size_t size, const U *rhs)`

- template<typename T, typename U>
bool [testing::internal::ArrayEq](#) (const T &lhs, const U &rhs)
- template<typename T, typename U, size_t N>
bool [testing::internal::ArrayEq](#) (const T(&lhs)[N], const U(&rhs)[N])
- template<typename Iter, typename Element>
Iter [testing::internal::ArrayAwareFind](#) (Iter begin, Iter end, const Element &elem)
- template<typename T, typename U>
void [testing::internal::CopyArray](#) (const T *from, size_t size, U *to)
- template<typename T, typename U>
void [testing::internal::CopyArray](#) (const T &from, U *to)
- template<typename T, typename U, size_t N>
void [testing::internal::CopyArray](#) (const T(&from)[N], U(*to)[N])

Zmienne

- [GTEST_API_ const char testing::internal::kStackTraceMarker \[\]](#)
- template<typename T>
bool [testing::internal::TypeidHelper< T >::dummy_](#) = false
- template<typename From, typename To>
const bool [testing::internal::ImplicitlyConvertible< From, To >::value](#)
- template<typename T>
const bool [testing::internal::IsHashTable< T >::value](#)

9.37.1 Dokumentacja definicji

9.37.1.1 GTEST_CONCAT_TOKEN_

```
#define GTEST_CONCAT_TOKEN_(  
    foo,  
    bar)
```

Wartość:

```
GTEST_CONCAT_TOKEN_IMPL_(foo, bar)
```

9.37.1.2 GTEST_CONCAT_TOKEN_IMPL_

```
#define GTEST_CONCAT_TOKEN_IMPL_(  
    foo,  
    bar)
```

Wartość:

```
foo ## bar
```

9.37.1.3 GTEST_FATAL_FAILURE_

```
#define GTEST_FATAL_FAILURE_(  
    message)
```

Wartość:

```
return GTEST\_MESSAGE\_\(message, ::testing::TestPartResult::kFatalFailure\)
```

9.37.1.4 GTEST_IS_NULL_LITERAL_

```
#define GTEST_IS_NULL_LITERAL_(  
    x)  
    (sizeof(::testing::internal::IsNullLiteralHelper(x)) == 1)
```

Wartość:

```
GTEST_IS_NULL_LITERAL_(__FILE__, __LINE__, message, result_type)
```

9.37.1.5 GTEST_MESSAGE_

```
#define GTEST_MESSAGE_(  
    message,  
    result_type)
```

Wartość:

```
GTEST_MESSAGE_AT_(__FILE__, __LINE__, message, result_type)
```

9.37.1.6 GTEST_MESSAGE_AT_

```
#define GTEST_MESSAGE_AT_(  
    file,  
    line,  
    message,  
    result_type)
```

Wartość:

```
::testing::internal::AssertHelper(result_type, file, line, message) \  
= ::testing::Message()
```

9.37.1.7 GTEST_NONFATAL_FAILURE_

```
#define GTEST_NONFATAL_FAILURE_(  
    message)
```

Wartość:

```
GTEST_MESSAGE_(message, ::testing::TestPartResult::kNonFatalFailure)
```

9.37.1.8 GTEST_REMOVE_CONST_

```
#define GTEST_REMOVE_CONST_(  
    T)
```

Wartość:

```
typename ::testing::internal::RemoveConst<T>::type
```

9.37.1.9 GTEST_REMOVE_REFERENCE_

```
#define GTEST_REMOVE_REFERENCE_(  
    T)
```

Wartość:

```
typename ::testing::internal::RemoveReference<T>::type
```

9.37.1.10 GTEST_REMOVE_REFERENCE_AND_CONST_

```
#define GTEST_REMOVE_REFERENCE_AND_CONST_(  
    T)
```

Wartość:

```
GTEST_REMOVE_CONST_(GTEST_REMOVE_REFERENCE_(T))
```

9.37.1.11 GTEST_STRINGIFY_

```
#define GTEST_STRINGIFY_(  
    name)
```

Wartość:

```
#name
```

9.37.1.12 GTEST_SUCCESS_

```
#define GTEST_SUCCESS_(  
    message)
```

Wartość:

```
GTEST_MESSAGE_(message, ::testing::TestPartResult::kSuccess)
```

9.37.1.13 GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING BELOW_

```
#define GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(  
    statement)
```

Wartość:

```
if (::testing::internal::AlwaysTrue()) { statement; }
```

9.37.1.14 GTEST_TEST_

```
#define GTEST_TEST_(  
    test_case_name,  
    test_name,  
    parent_class,  
    parent_id)
```

Wartość:

```
class GTEST_TEST_CLASS_NAME_(test_case_name, test_name) : public parent_class {\  
public:\  
    GTEST_TEST_CLASS_NAME_(test_case_name, test_name) () {} \  
private:\  
    virtual void TestBody(); \  
    static ::testing::TestInfo* const test_info_ GTEST_ATTRIBUTE_UNUSED_; \  
    GTEST_DISALLOW_COPY_AND_ASSIGN_(  
        GTEST_TEST_CLASS_NAME_(test_case_name, test_name)); \  
}; \  
\  
::testing::TestInfo* const GTEST_TEST_CLASS_NAME_(test_case_name, test_name) \  
    ::test_info_ = \  
        ::testing::internal::MakeAndRegisterTestInfo(\  
            #test_case_name, #test_name, NULL, NULL, \  
            ::testing::internal::CodeLocation(__FILE__, __LINE__), \  
            (parent_id), \  
            parent_class::SetUpTestCase, \  
            parent_class::TearDownTestCase, \  
            new ::testing::internal::TestFactoryImpl<\br/>                GTEST_TEST_CLASS_NAME_(test_case_name, test_name)>); \  
void GTEST_TEST_CLASS_NAME_(test_case_name, test_name)::TestBody()
```

9.37.1.15 GTEST_TEST_ANY_THROW_

```
#define GTEST_TEST_ANY_THROW_(  
    statement,  
    fail)
```

Wartość:

```
GTEST_AMBIGUOUS_ELSE_BLOCKER_ \  
if (::testing::internal::AlwaysTrue()) { \  
    bool gtest_caught_any = false; \  
    try { \  
        GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement); \  
    } \  
    catch (...) { \  
        gtest_caught_any = true; \  
    } \  
    if (!gtest_caught_any) { \  
        goto GTEST_CONCAT_TOKEN_(gtest_label_testanythrow_, __LINE__); \  
    } \  
} else \  
GTEST_CONCAT_TOKEN_(gtest_label_testanythrow_, __LINE__): \  
    fail("Expected: " #statement " throws an exception.\n" \  
         " Actual: it doesn't.")
```

9.37.1.16 GTEST_TEST_BOOLEAN_

```
#define GTEST_TEST_BOOLEAN_(  
    expression,  
    text,  
    actual,  
    expected,  
    fail)
```

Wartość:

```
GTEST_AMBIGUOUS_ELSE_BLOCKER_ \  
if (const ::testing::AssertionResult gtest_ar_ = \  
    ::testing::AssertionResult(expression)) \  
; \  
else \  
    fail(::testing::internal::GetBoolAssertionFailureMessage(\ \  
        gtest_ar_, text, #actual, #expected).c_str())
```

9.37.1.17 GTEST_TEST_CLASS_NAME_

```
#define GTEST_TEST_CLASS_NAME_(  
    test_case_name,  
    test_name)
```

Wartość:

```
test_case_name##_##test_name##_Test
```

9.37.1.18 GTEST_TEST_NO_FATAL_FAILURE_

```
#define GTEST_TEST_NO_FATAL_FAILURE_(  
    statement,  
    fail)
```

Wartość:

```
GTEST_AMBIGUOUS_ELSE_BLOCKER_ \  
if (::testing::internal::AlwaysTrue()) { \  
    ::testing::internal::HasNewFatalFailureHelper gtest_fatal_failure_checker; \  
    GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement); \  
    if (gtest_fatal_failure_checker.has_new_fatal_failure()) { \  
        goto GTEST_CONCAT_TOKEN_(gtest_label_testnofatal_, __LINE__); \  
    } \  
} else \  
GTEST_CONCAT_TOKEN_(gtest_label_testnofatal_, __LINE__): \  
    fail("Expected: " #statement " doesn't generate new fatal " \  
         "failures in the current thread.\n" \  
         " Actual: it does.")
```

9.37.1.19 GTEST_TEST_NO_THROW_

```
#define GTEST_TEST_NO_THROW_(  
    statement,  
    fail)
```

Wartość:

```
GTEST_AMBIGUOUS_ELSE_BLOCKER_ \  
if (::testing::internal::AlwaysTrue()) { \  
    try { \  
        GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement); \  
    } \  
    catch (...) { \  
        goto GTEST_CONCAT_TOKEN_(gtest_label_testnothrow_, __LINE__); \  
    } \  
} else { \  
    GTEST_CONCAT_TOKEN_(gtest_label_testnothrow_, __LINE__): \  
    fail("Expected: \"#statement\" doesn't throw an exception.\n" \  
         " Actual: it throws.")
```

9.37.1.20 GTEST_TEST_THROW_

```
#define GTEST_TEST_THROW_(  
    statement,  
    expected_exception,  
    fail)
```

Wartość:

```
GTEST_AMBIGUOUS_ELSE_BLOCKER_ \  
if (::testing::internal::ConstCharPtr gtest_msg = "") { \  
    bool gtest_caught_expected = false; \  
    try { \  
        GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement); \  
    } \  
    catch (expected_exception const&) { \  
        gtest_caught_expected = true; \  
    } \  
    catch (...) { \  
        gtest_msg.value = \  
            "Expected: \"#statement\" throws an exception of type \" \  
            #expected_exception \".\n Actual: it throws a different type.\"; \  
        goto GTEST_CONCAT_TOKEN_(gtest_label_testthrow_, __LINE__); \  
    } \  
    if (!gtest_caught_expected) { \  
        gtest_msg.value = \  
            "Expected: \"#statement\" throws an exception of type \" \  
            #expected_exception \".\n Actual: it throws nothing.\"; \  
        goto GTEST_CONCAT_TOKEN_(gtest_label_testthrow_, __LINE__); \  
    } \  
} else { \  
    GTEST_CONCAT_TOKEN_(gtest_label_testthrow_, __LINE__): \  
    fail(gtest_msg.value)
```

9.38 gtest-internal.h

[Idź do dokumentacji tego pliku.](#)

```
00001 // Copyright 2005, Google Inc.  
00002 // All rights reserved.  
00003 //  
00004 // Redistribution and use in source and binary forms, with or without  
00005 // modification, are permitted provided that the following conditions are  
00006 // met:  
00007 //  
00008 //      * Redistributions of source code must retain the above copyright  
00009 // notice, this list of conditions and the following disclaimer.  
00010 //      * Redistributions in binary form must reproduce the above  
00011 // copyright notice, this list of conditions and the following disclaimer  
00012 // in the documentation and/or other materials provided with the  
00013 // distribution.
```

```
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 // The Google C++ Testing and Mocking Framework (Google Test)
00031 //
00032 // This header file declares functions and macros used internally by
00033 // Google Test. They are subject to change without notice.
00034
00035 // GOOGLETST_CM0001 DO NOT DELETE
00036
00037 #ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_INTERNAL_H_
00038 #define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_INTERNAL_H_
00039
00040 #include "gtest/internal/gtest-port.h"
00041
00042 #if GTEST_OS_LINUX
00043 # include <stdlib.h>
00044 # include <sys/types.h>
00045 # include <sys/wait.h>
00046 # include <unistd.h>
00047 #endif // GTEST_OS_LINUX
00048
00049 #if GTEST_HAS_EXCEPTIONS
00050 # include <stdexcept>
00051 #endif
00052
00053 #include <ctype.h>
00054 #include <float.h>
00055 #include <string.h>
00056 #include <iomanip>
00057 #include <limits>
00058 #include <map>
00059 #include <set>
00060 #include <string>
00061 #include <vector>
00062
00063 #include "gtest/gtest-message.h"
00064 #include "gtest/internal/gtest-filepath.h"
00065 #include "gtest/internal/gtest-string.h"
00066 #include "gtest/internal/gtest-type-util.h"
00067
00068 // Due to C++ preprocessor weirdness, we need double indirection to
00069 // concatenate two tokens when one of them is __LINE__. Writing
00070 //
00071 // foo ## __LINE__
00072 //
00073 // will result in the token foo__LINE__, instead of foo followed by
00074 // the current line number. For more details, see
00075 // http://www.parashift.com/c++-faq-lite/misc-technical-issues.html#faq-39.6
00076 #define GTEST_CONCAT_TOKEN_(foo, bar) GTEST_CONCAT_TOKEN_IMPL_(foo, bar)
00077 #define GTEST_CONCAT_TOKEN_IMPL_(foo, bar) foo ## bar
00078
00079 // Stringifies its argument.
00080 #define GTEST_STRINGIFY_(name) #name
00081
00082 class ProtocolMessage;
00083 namespace proto2 { class Message; }
00084
00085 namespace testing {
00086
00087 // Forward declarations.
00088
00089 class AssertionResult;           // Result of an assertion.
00090 class Message;                 // Represents a failure message.
00091 class Test;                   // Represents a test.
00092 class TestInfo;                // Information about a test.
00093 class TestPartResult;          // Result of a test part.
00094 class UnitTest;                // A collection of test cases.
00095
00096 template <typename T>
00097 ::std::string PrintToString(const T& value);
00098
00099 namespace internal {
00100
```

```

00101 struct TraceInfo;                                // Information about a trace point.
00102 class TestInfoImpl;                            // Opaque implementation of TestInfo
00103 class UnitTestImpl;                           // Opaque implementation of UnitTest
00104
00105 // The text used in failure messages to indicate the start of the
00106 // stack trace.
00107 GTEST_API_ extern const char kStackTraceMarker[];
00108
00109 // Two overloaded helpers for checking at compile time whether an
00110 // expression is a null pointer literal (i.e. NULL or any 0-valued
00111 // compile-time integral constant). Their return values have
00112 // different sizes, so we can use sizeof() to test which version is
00113 // picked by the compiler. These helpers have no implementations, as
00114 // we only need their signatures.
00115 //
00116 // Given IsNullLiteralHelper(x), the compiler will pick the first
00117 // version if x can be implicitly converted to Secret*, and pick the
00118 // second version otherwise. Since Secret is a secret and incomplete
00119 // type, the only expression a user can write that has type Secret* is
00120 // a null pointer literal. Therefore, we know that x is a null
00121 // pointer literal if and only if the first version is picked by the
00122 // compiler.
00123 char IsNullLiteralHelper(Secret* p);
00124 char (&IsNullLiteralHelper(...))[2]; // NOLINT
00125
00126 // A compile-time bool constant that is true if and only if x is a
00127 // null pointer literal (i.e. NULL or any 0-valued compile-time
00128 // integral constant).
00129 #ifdef GTEST_ELLIPSIS_NEEDS_POD_
00130 // We lose support for NULL detection where the compiler doesn't like
00131 // passing non-POD classes through ellipsis (...).
00132 # define GTEST_IS_NULL_LITERAL_(x) false
00133 #else
00134 # define GTEST_IS_NULL_LITERAL_(x) \
00135     (sizeof(::testing::internal::IsNullLiteralHelper(x)) == 1)
00136 #endif // GTEST_ELLIPSIS_NEEDS_POD_
00137
00138 // Appends the user-supplied message to the Google-Test-generated message.
00139 GTEST_API_ std::string AppendUserMessage(
00140     const std::string& gtest_msg, const Message& user_msg);
00141
00142 #if GTEST_HAS_EXCEPTIONS
00143
00144 GTEST_DISABLE_MSC_WARNINGS_PUSH_(4275 \
00145 /* an exported class was derived from a class that was not exported */)
00146
00147 // This exception is thrown by (and only by) a failed Google Test
00148 // assertion when GTEST_FLAG(throw_on_failure) is true (if exceptions
00149 // are enabled). We derive it from std::runtime_error, which is for
00150 // errors presumably detectable only at run time. Since
00151 // std::runtime_error inherits from std::exception, many testing
00152 // frameworks know how to extract and print the message inside it.
00153 class GTEST_API_ GoogleTestFailureException : public ::std::runtime_error {
00154 public:
00155     explicit GoogleTestFailureException(const TestPartResult& failure);
00156 };
00157
00158 GTEST_DISABLE_MSC_WARNINGS_POP_() // 4275
00159
00160 #endif // GTEST_HAS_EXCEPTIONS
00161
00162 namespace edit_distance {
00163 // Returns the optimal edits to go from 'left' to 'right'.
00164 // All edits cost the same, with replace having lower priority than
00165 // add/remove.
00166 // Simple implementation of the Wagner-Fischer algorithm.
00167 // See http://en.wikipedia.org/wiki/Wagner-Fischer_algorithm
00168 enum EditType { kMatch, kAdd, kRemove, kReplace };
00169 GTEST_API_ std::vector<EditType> CalculateOptimalEdits(
00170     const std::vector<size_t>& left, const std::vector<size_t>& right);
00171
00172 // Same as above, but the input is represented as strings.
00173 GTEST_API_ std::vector<EditType> CalculateOptimalEdits(
00174     const std::vector<std::string>& left,
00175     const std::vector<std::string>& right);
00176
00177 // Create a diff of the input strings in Unified diff format.
00178 GTEST_API_ std::string CreateUnifiedDiff(const std::vector<std::string>& left,
00179                                         const std::vector<std::string>& right,
00180                                         size_t context = 2);
00181
00182 } // namespace edit_distance
00183
00184 // Calculate the diff between 'left' and 'right' and return it in unified diff
00185 // format.
00186 // If not null, stores in 'total_line_count' the total number of lines found
00187 // in left + right.

```

```

00188 GTEST_API_ std::string DiffStrings(const std::string& left,
00189                                     const std::string& right,
00190                                     size_t* total_line_count);
00191
00192 // Constructs and returns the message for an equality assertion
00193 // (e.g. ASSERT_EQ, EXPECT_STREQ, etc) failure.
00194 //
00195 // The first four parameters are the expressions used in the assertion
00196 // and their values, as strings. For example, for ASSERT_EQ(foo, bar)
00197 // where foo is 5 and bar is 6, we have:
00198 //
00199 //   expected_expression: "foo"
00200 //   actual_expression:   "bar"
00201 //   expected_value:     "5"
00202 //   actual_value:       "6"
00203 //
00204 // The ignoring_case parameter is true iff the assertion is a
00205 // *_STRCASEEQ*. When it's true, the string "(ignoring case)" will
00206 // be inserted into the message.
00207 GTEST_API_ AssertionResult EqFailure(const char* expected_expression,
00208                                         const char* actual_expression,
00209                                         const std::string& expected_value,
00210                                         const std::string& actual_value,
00211                                         bool ignoring_case);
00212
00213 // Constructs a failure message for Boolean assertions such as EXPECT_TRUE.
00214 GTEST_API_ std::string GetBoolAssertionFailureMessage(
00215     const AssertionResult& assertion_result,
00216     const char* expression_text,
00217     const char* actual_predicate_value,
00218     const char* expected_predicate_value);
00219
00220 // This template class represents an IEEE floating-point number
00221 // (either single-precision or double-precision, depending on the
00222 // template parameters).
00223 //
00224 // The purpose of this class is to do more sophisticated number
00225 // comparison. (Due to round-off error, etc, it's very unlikely that
00226 // two floating-points will be equal exactly. Hence a naive
00227 // comparison by the == operation often doesn't work.)
00228 //
00229 // Format of IEEE floating-point:
00230 //
00231 //   The most-significant bit being the leftmost, an IEEE
00232 //   floating-point looks like
00233 //
00234 //     sign_bit exponent_bits fraction_bits
00235 //
00236 //   Here, sign_bit is a single bit that designates the sign of the
00237 //   number.
00238 //
00239 //   For float, there are 8 exponent bits and 23 fraction bits.
00240 //
00241 //   For double, there are 11 exponent bits and 52 fraction bits.
00242 //
00243 //   More details can be found at
00244 //   http://en.wikipedia.org/wiki/IEEE_floating-point_standard.
00245 //
00246 // Template parameter:
00247 //
00248 //   RawType: the raw floating-point type (either float or double)
00249 template <typename RawType>
00250 class FloatingPoint {
00251 public:
00252     // Defines the unsigned integer type that has the same size as the
00253     // floating point number.
00254     typedef typename TypeWithSize<sizeof(RawType)>::UInt Bits;
00255
00256     // Constants.
00257
00258     // # of bits in a number.
00259     static const size_t kBitCount = 8*sizeof(RawType);
00260
00261     // # of fraction bits in a number.
00262     static const size_t kFractionBitCount =
00263         std::numeric_limits<RawType>::digits - 1;
00264
00265     // # of exponent bits in a number.
00266     static const size_t kExponentBitCount = kBitCount - 1 - kFractionBitCount;
00267
00268     // The mask for the sign bit.
00269     static const Bits kSignBitMask = static_cast<Bits>(1) << (kBitCount - 1);
00270
00271     // The mask for the fraction bits.
00272     static const Bits kFractionBitMask =
00273         ~static_cast<Bits>(0) >> (kExponentBitCount + 1);
00274

```

```
00275 // The mask for the exponent bits.
00276 static const Bits kExponentBitMask = ~(kSignBitMask | kFractionBitMask);
00277
00278 // How many ULP's (Units in the Last Place) we want to tolerate when
00279 // comparing two numbers. The larger the value, the more error we
00280 // allow. A 0 value means that two numbers must be exactly the same
00281 // to be considered equal.
00282 //
00283 // The maximum error of a single floating-point operation is 0.5
00284 // units in the last place. On Intel CPU's, all floating-point
00285 // calculations are done with 80-bit precision, while double has 64
00286 // bits. Therefore, 4 should be enough for ordinary use.
00287 //
00288 // See the following article for more details on ULP:
00289 // http://randomascii.wordpress.com/2012/02/25/comparing-floating-point-numbers-2012-edition/
00290 static const size_t kMaxUlps = 4;
00291
00292 // Constructs a FloatingPoint from a raw floating-point number.
00293 //
00294 // On an Intel CPU, passing a non-normalized NAN (Not a Number)
00295 // around may change its bits, although the new value is guaranteed
00296 // to be also a NAN. Therefore, don't expect this constructor to
00297 // preserve the bits in x when x is a NAN.
00298 explicit FloatingPoint(const RawType& x) { u_.value_ = x; }
00299
00300 // Static methods
00301
00302 // Reinterprets a bit pattern as a floating-point number.
00303 //
00304 // This function is needed to test the AlmostEquals() method.
00305 static RawType ReinterpretBits(const Bits bits) {
00306     FloatingPoint fp(0);
00307     fp.u_.bits_ = bits;
00308     return fp.u_.value_;
00309 }
00310
00311 // Returns the floating-point number that represent positive infinity.
00312 static RawType Infinity() {
00313     return ReinterpretBits(kExponentBitMask);
00314 }
00315
00316 // Returns the maximum representable finite floating-point number.
00317 static RawType Max();
00318
00319 // Non-static methods
00320
00321 // Returns the bits that represents this number.
00322 const Bits &bits() const { return u_.bits_; }
00323
00324 // Returns the exponent bits of this number.
00325 Bits exponent_bits() const { return kExponentBitMask & u_.bits_; }
00326
00327 // Returns the fraction bits of this number.
00328 Bits fraction_bits() const { return kFractionBitMask & u_.bits_; }
00329
00330 // Returns the sign bit of this number.
00331 Bits sign_bit() const { return kSignBitMask & u_.bits_; }
00332
00333 // Returns true iff this is NAN (not a number).
00334 bool is_nan() const {
00335     // It's a NAN if the exponent bits are all ones and the fraction
00336     // bits are not entirely zeros.
00337     return (exponent_bits() == kExponentBitMask) && (fraction_bits() != 0);
00338 }
00339
00340 // Returns true iff this number is at most kMaxUlps ULP's away from
00341 // rhs. In particular, this function:
00342 //
00343 // - returns false if either number is (or both are) NAN.
00344 // - treats really large numbers as almost equal to infinity.
00345 // - thinks +0.0 and -0.0 are 0 DLP's apart.
00346 bool AlmostEquals(const FloatingPoint& rhs) const {
00347     // The IEEE standard says that any comparison operation involving
00348     // a NAN must return false.
00349     if (is_nan() || rhs.is_nan()) return false;
00350
00351     return DistanceBetweenSignAndMagnitudeNumbers(u_.bits_, rhs.u_.bits_)
00352         <= kMaxUlps;
00353 }
00354
00355 private:
00356 // The data type used to store the actual floating-point number.
00357 union FloatingPointUnion {
00358     RawType value_; // The raw floating-point number.
00359     Bits bits_; // The bits that represent the number.
00360 };
00361
```

```

00362 // Converts an integer from the sign-and-magnitude representation to
00363 // the biased representation. More precisely, let N be 2 to the
00364 // power of (kBitCount - 1), an integer x is represented by the
00365 // unsigned number x + N.
00366 //
00367 // For instance,
00368 //
00369 // -N + 1 (the most negative number representable using
00370 // sign-and-magnitude) is represented by 1;
00371 // 0 is represented by N; and
00372 // N - 1 (the biggest number representable using
00373 // sign-and-magnitude) is represented by 2N - 1.
00374 //
00375 // Read http://en.wikipedia.org/wiki/Signed_number_representations
00376 // for more details on signed number representations.
00377 static Bits SignAndMagnitudeToBiased(const Bits &sam) {
00378     if (kSignBitMask & sam) {
00379         // sam represents a negative number.
00380         return ~sam + 1;
00381     } else {
00382         // sam represents a positive number.
00383         return kSignBitMask | sam;
00384     }
00385 }
00386
00387 // Given two numbers in the sign-and-magnitude representation,
00388 // returns the distance between them as an unsigned number.
00389 static Bits DistanceBetweenSignAndMagnitudeNumbers(const Bits &saml,
00390                                         const Bits &sam2) {
00391     const Bits biased1 = SignAndMagnitudeToBiased(saml);
00392     const Bits biased2 = SignAndMagnitudeToBiased(sam2);
00393     return (biased1 >= biased2) ? (biased1 - biased2) : (biased2 - biased1);
00394 }
00395
00396 FloatingPointUnion u_;
00397 };
00398
00399 // We cannot use std::numeric_limits<T>::max() as it clashes with the max()
00400 // macro defined by <windows.h>.
00401 template <>
00402 inline float FloatingPoint<float>::Max() { return FLT_MAX; }
00403 template <>
00404 inline double FloatingPoint<double>::Max() { return DBL_MAX; }
00405
00406 // Typedefs the instances of the FloatingPoint template class that we
00407 // care to use.
00408 typedef FloatingPoint<float> Float;
00409 typedef FloatingPoint<double> Double;
00410
00411 // In order to catch the mistake of putting tests that use different
00412 // test fixture classes in the same test case, we need to assign
00413 // unique IDs to fixture classes and compare them. The TypeId type is
00414 // used to hold such IDs. The user should treat TypeId as an opaque
00415 // type: the only operation allowed on TypeId values is to compare
00416 // them for equality using the == operator.
00417 typedef const void* TypeId;
00418
00419 template <typename T>
00420 class TypeIdHelper {
00421 public:
00422     // dummy_ must not have a const type. Otherwise an overly eager
00423     // compiler (e.g. MSVC 7.1 & 8.0) may try to merge
00424     // TypeIdHelper<T>::dummy_ for different Ts as an "optimization".
00425     static bool dummy_;
00426 };
00427
00428 template <typename T>
00429 bool TypeIdHelper<T>::dummy_ = false;
00430
00431 // GetTypeId<T>() returns the ID of type T. Different values will be
00432 // returned for different types. Calling the function twice with the
00433 // same type argument is guaranteed to return the same ID.
00434 template <typename T>
00435 TypeId GetTypeId() {
00436     // The compiler is required to allocate a different
00437     // TypeIdHelper<T>::dummy_ variable for each T used to instantiate
00438     // the template. Therefore, the address of dummy_ is guaranteed to
00439     // be unique.
00440     return &(TypeIdHelper<T>::dummy_);
00441 }
00442
00443 // Returns the type ID of ::testing::Test. Always call this instead
00444 // of GetTypeId< ::testing::Test>() to get the type ID of
00445 // ::testing::Test, as the latter may give the wrong result due to a
00446 // suspected linker bug when compiling Google Test as a Mac OS X
00447 // framework.
00448 GTEST_API_ TypeId GetTestTypeId();

```

```
00449
00450 // Defines the abstract factory interface that creates instances
00451 // of a Test object.
00452 class TestFactoryBase {
00453 public:
00454     virtual ~TestFactoryBase() {}
00455
00456     // Creates a test instance to run. The instance is both created and destroyed
00457     // within TestInfoImpl::Run()
00458     virtual Test* CreateTest() = 0;
00459
00460 protected:
00461     TestFactoryBase() {}
00462
00463 private:
00464     GTEST_DISALLOW_COPY_AND_ASSIGN_(TestFactoryBase);
00465 };
00466
00467 // This class provides implementation of TeastFactoryBase interface.
00468 // It is used in TEST and TEST_F macros.
00469 template <class TestClass>
00470 class TestFactoryImpl : public TestFactoryBase {
00471 public:
00472     virtual Test* CreateTest() { return new TestClass; }
00473 };
00474
00475 #if GTEST_OS_WINDOWS
00476
00477 // Predicate-formatters for implementing the HRESULT checking macros
00478 // {ASSERT|EXPECT}_HRESULT_{SUCCEEDED|FAILED}
00479 // We pass a long instead of HRESULT to avoid causing an
00480 // include dependency for the HRESULT type.
00481 GTEST_API_ AssertionResult IsHRESULTSuccess(const char* expr,
00482                                              long hr); // NOLINT
00483 GTEST_API_ AssertionResult IsHRESULTFailure(const char* expr,
00484                                              long hr); // NOLINT
00485
00486 #endif // GTEST_OS_WINDOWS
00487
00488 // Types of SetUpTestCase() and TearDownTestCase() functions.
00489 typedef void (*SetUpTestCaseFunc)();
00490 typedef void (*TearDownTestCaseFunc)();
00491
00492 struct CodeLocation {
00493     CodeLocation(const std::string& a_file, int a_line)
00494         : file(a_file), line(a_line) {}
00495
00496     std::string file;
00497     int line;
00498 };
00499
00500 // Creates a new TestInfo object and registers it with Google Test;
00501 // returns the created object.
00502 //
00503 // Arguments:
00504 //
00505 // test_case_name: name of the test case
00506 // name: name of the test
00507 // type_param the name of the test's type parameter, or NULL if
00508 // this is not a typed or a type-parameterized test.
00509 // value_param text representation of the test's value parameter,
00510 // or NULL if this is not a type-parameterized test.
00511 // code_location: code location where the test is defined
00512 // fixture_class_id: ID of the test fixture class
00513 // set_up_tc: pointer to the function that sets up the test case
00514 // tear_down_tc: pointer to the function that tears down the test case
00515 // factory: pointer to the factory that creates a test object.
00516 // The newly created TestInfo instance will assume
00517 // ownership of the factory object.
00518 GTEST_API_ TestInfo* MakeAndRegisterTestInfo(
00519     const char* test_case_name,
00520     const char* name,
00521     const char* type_param,
00522     const char* value_param,
00523     CodeLocation code_location,
00524     TypeId fixture_class_id,
00525     SetUpTestCaseFunc set_up_tc,
00526     TearDownTestCaseFunc tear_down_tc,
00527     TestFactoryBase* factory);
00528
00529 // If *pstr starts with the given prefix, modifies *pstr to be right
00530 // past the prefix and returns true; otherwise leaves *pstr unchanged
00531 // and returns false. None of pstr, *pstr, and prefix can be NULL.
00532 GTEST_API_ bool SkipPrefix(const char* prefix, const char** pstr);
00533
00534 #if GTEST_HAS_TYPED_TEST || GTEST_HAS_TYPED_TEST_P
00535
```

```
00536 GTEST_DISABLE_MSC_WARNINGS_PUSH_(4251 \
00537 /* class A needs to have dll-interface to be used by clients of class B */
00538
00539 // State of the definition of a type-parameterized test case.
00540 class GTEST_API_ TypedTestCasePState {
00541 public:
00542 TypedTestCasePState() : registered_(false) {}
00543
00544 // Adds the given test name to defined_test_names_ and return true
00545 // if the test case hasn't been registered; otherwise aborts the
00546 // program.
00547 bool AddTestName(const char* file, int line, const char* case_name,
00548 const char* test_name) {
00549 if (registered_) {
00550 fprintf(stderr, "%s Test %s must be defined before "
00551 "REGISTER_TYPED_TEST_CASE_P(%s, ...).\n",
00552 FormatFileLocation(file, line).c_str(), test_name, case_name);
00553 fflush(stderr);
00554 posix::Abort();
00555 }
00556 registered_tests_.insert(
00557 ::std::make_pair(test_name, CodeLocation(file, line)));
00558 return true;
00559 }
00560
00561 bool TestExists(const std::string& test_name) const {
00562 return registered_tests_.count(test_name) > 0;
00563 }
00564
00565 const CodeLocation& GetCodeLocation(const std::string& test_name) const {
00566 RegisteredTestsMap::const_iterator it = registered_tests_.find(test_name);
00567 GTEST_CHECK_(it != registered_tests_.end());
00568 return it->second;
00569 }
00570
00571 // Verifies that registered_tests match the test names in
00572 // defined_test_names_; returns registered_tests if successful, or
00573 // aborts the program otherwise.
00574 const char* VerifyRegisteredTestNames(
00575 const char* file, int line, const char* registered_tests);
00576
00577 private:
00578 typedef ::std::map<std::string, CodeLocation> RegisteredTestsMap;
00579
00580 bool registered_;
00581 RegisteredTestsMap registered_tests_;
00582 };
00583
00584 GTEST_DISABLE_MSC_WARNINGS_POP_() // 4251
00585
00586 // Skips to the first non-space char after the first comma in 'str';
00587 // returns NULL if no comma is found in 'str'.
00588 inline const char* SkipComma(const char* str) {
00589 const char* comma = strchr(str, ',');
00590 if (comma == NULL) {
00591 return NULL;
00592 }
00593 while (IsSpace(*(++comma))) {}
00594 return comma;
00595 }
00596
00597 // Returns the prefix of 'str' before the first comma in it; returns
00598 // the entire string if it contains no comma.
00599 inline std::string GetPrefixUntilComma(const char* str) {
00600 const char* comma = strchr(str, ',');
00601 return comma == NULL ? str : std::string(str, comma);
00602 }
00603
00604 // Splits a given string on a given delimiter, populating a given
00605 // vector with the fields.
00606 void SplitString(const ::std::string& str, char delimiter,
00607 ::std::vector< ::std::string>* dest);
00608
00609 // The default argument to the template below for the case when the user does
00610 // not provide a name generator.
00611 struct DefaultNameGenerator {
00612 template <typename T>
00613 static std::string GetName(int i) {
00614 return StreamableToString(i);
00615 }
00616 };
00617
00618 template <typename Provided = DefaultNameGenerator>
00619 struct NameGeneratorSelector {
00620 typedef Provided type;
00621 };
00622
```

```
00623 template <typename NameGenerator>
00624 void GenerateNamesRecursively(Types0, std::vector<std::string>*, int) {}
00625
00626 template <typename NameGenerator, typename Types>
00627 void GenerateNamesRecursively(Types, std::vector<std::string>* result, int i) {
00628     result->push_back(NameGenerator::template GetName<typename Types::Head>(i));
00629     GenerateNamesRecursively<NameGenerator>(typename Types::Tail(), result,
00630                                         i + 1);
00631 }
00632
00633 template <typename NameGenerator, typename Types>
00634 std::vector<std::string> GenerateNames() {
00635     std::vector<std::string> result;
00636     GenerateNamesRecursively<NameGenerator>(Types(), &result, 0);
00637     return result;
00638 }
00639
00640 // TypeParameterizedTest<Fixture, TestSel, Types>::Register()
00641 // registers a list of type-parameterized tests with Google Test. The
00642 // return value is insignificant - we just need to return something
00643 // such that we can call this function in a namespace scope.
00644 //
00645 // Implementation note: The GTEST_TEMPLATE_ macro declares a template
00646 // template parameter. It's defined in gtest-type-util.h.
00647 template <GTEST_TEMPLATE_ Fixture, class TestSel, typename Types>
00648 class TypeParameterizedTest {
00649 public:
00650     // 'index' is the index of the test in the type list 'Types'
00651     // specified in INSTANTIATE_TYPED_TEST_CASE_P(Prefix, TestCase,
00652     // Types). Valid values for 'index' are [0, N - 1] where N is the
00653     // length of Types.
00654     static bool Register(const char* prefix, const CodeLocation& code_location,
00655                          const char* case_name, const char* test_names, int index,
00656                          const std::vector<std::string>& type_names =
00657                                      GenerateNames<DefaultNameGenerator, Types>()) {
00658         typedef typename Types::Head Type;
00659         typedef Fixture<Type> FixtureClass;
00660         typedef typename GTEST_BIND_(TestSel, Type) TestClass;
00661
00662         // First, registers the first type-parameterized test in the type
00663         // list.
00664         MakeAndRegisterTestInfo(
00665             (std::string(prefix) + (prefix[0] == '\0' ? "" : "/") + case_name +
00666             "/" + type_names[index])
00667             .c_str(),
00668             StripTrailingSpaces(GetPrefixUntilComma(test_names)).c_str(),
00669             GetTypeName<Type>().c_str(),
00670             NULL, // No value parameter.
00671             code_location, GetTypeId<FixtureClass>(), TestClass::SetUpTestCase,
00672             TestClass::TearDownTestCase, new TestFactoryImpl<TestClass>());
00673
00674         // Next, recurses (at compile time) with the tail of the type list.
00675         return TypeParameterizedTest<Fixture, TestSel,
00676                               typename Types::Tail>::Register(prefix,
00677                                         code_location,
00678                                         case_name,
00679                                         test_names,
00680                                         index + 1,
00681                                         type_names);
00682     }
00683 };
00684
00685 // The base case for the compile time recursion.
00686 template <GTEST_TEMPLATE_ Fixture, class TestSel>
00687 class TypeParameterizedTest<Fixture, TestSel, Types0> {
00688 public:
00689     static bool Register(const char* /*prefix*/, const CodeLocation&,
00690                          const char* /*case_name*/, const char* /*test_names*/,
00691                          int /*index*/,
00692                          const std::vector<std::string>& =
00693                          std::vector<std::string>() /*type_names*/) {
00694         return true;
00695     }
00696 };
00697
00698 // TypeParameterizedTestCase<Fixture, Tests, Types>::Register()
00699 // registers *all* combinations* of 'Tests' and 'Types' with Google
00700 // Test. The return value is insignificant - we just need to return
00701 // something such that we can call this function in a namespace scope.
00702 template <GTEST_TEMPLATE_ Fixture, typename Tests, typename Types>
00703 class TypeParameterizedTestCase {
00704 public:
00705     static bool Register(const char* prefix, CodeLocation code_location,
00706                          const TypedTestCasePState* state, const char* case_name,
00707                          const char* test_names,
00708                          const std::vector<std::string>& type_names =
00709                          GenerateNames<DefaultNameGenerator, Types>()) {
```

```

00710     std::string test_name = StripTrailingSpaces(
00711         GetPrefixUntilComma(test_names));
00712     if (!state->TestExists(test_name)) {
00713         fprintf(stderr, "Failed to get code location for test %s.%s at %s.",
00714             case_name, test_name.c_str(),
00715             FormatFileLocation(code_location.file.c_str(),
00716                 code_location.line).c_str());
00717         fflush(stderr);
00718         posix::Abort();
00719     }
00720     const CodeLocation& test_location = state->GetCodeLocation(test_name);
00721
00722     typedef typename Tests::Head Head;
00723
00724     // First, register the first test in 'Test' for each type in 'Types'.
00725     TypeParameterizedTest<Fixture, Head, Types>::Register(
00726         prefix, test_location, case_name, test_names, 0, type_names);
00727
00728     // Next, recurses (at compile time) with the tail of the test list.
00729     return TypeParameterizedTestCase<Fixture, typename Tests::Tail,
00730         Types>::Register(prefix, code_location,
00731                         state, case_name,
00732                         SkipComma(test_names),
00733                         type_names);
00734 }
00735 };
00736
00737 // The base case for the compile time recursion.
00738 template <GTEST_TEMPLATE_ Fixture, typename Types>
00739 class TypeParameterizedTestCase<Fixture, Templates0, Types> {
00740     public:
00741     static bool Register(const char* /*prefix*/, const CodeLocation&,
00742         const TypedTestCasePState* /*state*/,
00743         const char* /*case_name*/, const char* /*test_names*/,
00744         const std::vector<std::string>& =
00745             std::vector<std::string>() /*type_names*/) {
00746     return true;
00747 }
00748 };
00749
00750 #endif // GTEST_HAS_TYPED_TEST || GTEST_HAS_TYPED_TEST_P
00751
00752 // Returns the current OS stack trace as an std::string.
00753 //
00754 // The maximum number of stack frames to be included is specified by
00755 // the gtest_stack_trace_depth flag. The skip_count parameter
00756 // specifies the number of top frames to be skipped, which doesn't
00757 // count against the number of frames to be included.
00758 //
00759 // For example, if Foo() calls Bar(), which in turn calls
00760 // GetCurrentOsStackTraceExceptTop(..., 1), Foo() will be included in
00761 // the trace but Bar() and GetCurrentOsStackTraceExceptTop() won't.
00762 GTEST_API_ std::string GetCurrentOsStackTraceExceptTop(
00763     UnitTest* unit_test, int skip_count);
00764
00765 // Helpers for suppressing warnings on unreachable code or constant
00766 // condition.
00767
00768 // Always returns true.
00769 GTEST_API_ bool AlwaysTrue();
00770
00771 // Always returns false.
00772 inline bool AlwaysFalse() { return !AlwaysTrue(); }
00773
00774 // Helper for suppressing false warning from Clang on a const char*
00775 // variable declared in a conditional expression always being NULL in
00776 // the else branch.
00777 struct GTEST_API_ ConstCharPtr {
00778     ConstCharPtr(const char* str) : value(str) {}
00779     operator bool() const { return true; }
00780     const char* value;
00781 };
00782
00783 // A simple Linear Congruential Generator for generating random
00784 // numbers with a uniform distribution. Unlike rand() and srand(), it
00785 // doesn't use global state (and therefore can't interfere with user
00786 // code). Unlike rand_r(), it's portable. An LCG isn't very random,
00787 // but it's good enough for our purposes.
00788 class GTEST_API_ Random {
00789     public:
00790     static const UInt32 kMaxRange = lu « 31;
00791
00792     explicit Random(UInt32 seed) : state_(seed) {}
00793
00794     void Reseed(UInt32 seed) { state_ = seed; }
00795
00796     // Generates a random number from [0, range). Crashes if 'range' is

```

```
00797 // 0 or greater than kMaxRange.
00798 UInt32 Generate(UInt32 range);
00799
00800 private:
00801 UInt32 state_;
00802 GTEST_DISALLOW_COPY_AND_ASSIGN_(Random);
00803 };
00804
00805 // Defining a variable of type CompileAssertTypesEqual<T1, T2> will cause a
00806 // compiler error iff T1 and T2 are different types.
00807 template <typename T1, typename T2>
00808 struct CompileAssertTypesEqual;
00809
00810 template <typename T>
00811 struct CompileAssertTypesEqual<T, T> {
00812 };
00813
00814 // Removes the reference from a type if it is a reference type,
00815 // otherwise leaves it unchanged. This is the same as
00816 // tr1::remove_reference, which is not widely available yet.
00817 template <typename T>
00818 struct RemoveReference { typedef T type; }; // NOLINT
00819 template <typename T>
00820 struct RemoveReference<T&> { typedef T type; }; // NOLINT
00821
00822 // A handy wrapper around RemoveReference that works when the argument
00823 // T depends on template parameters.
00824 #define GTEST_REMOVE_REFERENCE_(T) \
00825     typename ::testing::internal::RemoveReference<T>::type
00826
00827 // Removes const from a type if it is a const type, otherwise leaves
00828 // it unchanged. This is the same as tr1::remove_const, which is not
00829 // widely available yet.
00830 template <typename T>
00831 struct RemoveConst { typedef T type; }; // NOLINT
00832 template <typename T>
00833 struct RemoveConst<const T> { typedef T type; }; // NOLINT
00834
00835 // MSVC 8.0, Sun C++, and IBM XL C++ have a bug which causes the above
00836 // definition to fail to remove the const in 'const int[3]' and 'const
00837 // char[3][4]'. The following specialization works around the bug.
00838 template <typename T, size_t N>
00839 struct RemoveConst<const T[N]> {
00840     typedef typename RemoveConst<T>::type type[N];
00841 };
00842
00843 #if defined(_MSC_VER) && _MSC_VER < 1400
00844 // This is the only specialization that allows VC++ 7.1 to remove const in
00845 // 'const int[3]' and 'const int[3][4]'. However, it causes trouble with GCC
00846 // and thus needs to be conditionally compiled.
00847 template <typename T, size_t N>
00848 struct RemoveConst<T[N]> {
00849     typedef typename RemoveConst<T>::type type[N];
00850 };
00851 #endif
00852
00853 // A handy wrapper around RemoveConst that works when the argument
00854 // T depends on template parameters.
00855 #define GTEST_REMOVE_CONST_(T) \
00856     typename ::testing::internal::RemoveConst<T>::type
00857
00858 // Turns const U&, U&, const U, and U all into U.
00859 #define GTEST_REMOVE_REFERENCE_AND_CONST_(T) \
00860     GTEST_REMOVE_CONST_(GTEST_REMOVE_REFERENCE_(T))
00861
00862 // ImplicitlyConvertible<From, To>::value is a compile-time bool
00863 // constant that's true iff type From can be implicitly converted to
00864 // type To.
00865 template <typename From, typename To>
00866 class ImplicitlyConvertible {
00867 private:
00868     // We need the following helper functions only for their types.
00869     // They have no implementations.
00870
00871     // MakeFrom() is an expression whose type is From. We cannot simply
00872     // use From(), as the type From may not have a public default
00873     // constructor.
00874     static typename AddReference<From>::type MakeFrom();
00875
00876     // These two functions are overloaded. Given an expression
00877     // Helper(x), the compiler will pick the first version if x can be
00878     // implicitly converted to type To; otherwise it will pick the
00879     // second version.
00880     //
00881     // The first version returns a value of size 1, and the second
00882     // version returns a value of size 2. Therefore, by checking the
00883     // size of Helper(x), which can be done at compile time, we can tell
```

```

0084 // which version of Helper() is used, and hence whether x can be
0085 // implicitly converted to type To.
0086 static char Helper(To);
0087 static char (&Helper(...))[2]; // NOLINT
0088
0089 // We have to put the 'public' section after the 'private' section,
0090 // or MSVC refuses to compile the code.
0091 public:
0092 #if defined(__BORLANDC__)
0093 // C++Builder cannot use member overload resolution during template
0094 // instantiation. The simplest workaround is to use its C++0x type traits
0095 // functions (C++Builder 2009 and above only).
0096 static const bool value = __is_convertible(From, To);
0097 #else
0098 // MSVC warns about implicitly converting from double to int for
0099 // possible loss of data, so we need to temporarily disable the
0100 // warning.
0101 GTEST_DISABLE_MSC_WARNINGS_PUSH_(4244)
0102 static const bool value =
0103     sizeof(Helper(ImplicitlyConvertible::MakeFrom())) == 1;
0104 GTEST_DISABLE_MSC_WARNINGS_POP_()
0105 #endif // __BORLANDC__
0106 };
0107 template <typename From, typename To>
0108 const bool ImplicitlyConvertible<From, To>::value;
0109
0110 // IsAProtocolMessage<T>::value is a compile-time bool constant that's
0111 // true iff T is type ProtocolMessage, proto2::Message, or a subclass
0112 // of those.
0113 template <typename T>
0114 struct IsAProtocolMessage
0115 : public bool_constant<
0116     ImplicitlyConvertible<const T*, const ::ProtocolMessage*>::value ||
0117     ImplicitlyConvertible<const T*, const ::proto2::Message*>::value> {
0118 };
0119
0120 // When the compiler sees expression IsContainerTest<C>(0), if C is an
0121 // STL-style container class, the first overload of IsContainerTest
0122 // will be viable (since both C::iterator* and C::const_iterator* are
0123 // valid types and NULL can be implicitly converted to them). It will
0124 // be picked over the second overload as 'int' is a perfect match for
0125 // the type of argument 0. If C::iterator or C::const_iterator is not
0126 // a valid type, the first overload is not viable, and the second
0127 // overload will be picked. Therefore, we can determine whether C is
0128 // a container class by checking the type of IsContainerTest<C>(0).
0129 // The value of the expression is insignificant.
0130 //
0131 // In C++11 mode we check the existence of a const_iterator and that an
0132 // iterator is properly implemented for the container.
0133 //
0134 // For pre-C++11 that we look for both C::iterator and C::const_iterator.
0135 // The reason is that C++ injects the name of a class as a member of the
0136 // class itself (e.g. you can refer to class iterator as either
0137 // 'iterator' or 'iterator::iterator'). If we look for C::iterator
0138 // only, for example, we would mistakenly think that a class named
0139 // iterator is an STL container.
0140 //
0141 // Also note that the simpler approach of overloading
0142 // IsContainerTest(typename C::const_iterator*) and
0143 // IsContainerTest(...) doesn't work with Visual Age C++ and Sun C++.
0144 typedef int IsContainer;
0145 #if GTEST_LANG_CXX11
0146 template <class C,
0147         class Iterator = decltype(::std::declval<const C&>().begin()),
0148         class = decltype(::std::declval<const C&>().end()),
0149         class = decltype(++::std::declval<Iterator&>()),
0150         class = decltype(*::std::declval<Iterator>()),
0151         class = typename C::const_iterator>
0152 IsContainer IsContainerTest(int /* dummy */) {
0153     return 0;
0154 }
0155 #else
0156 template <class C>
0157 IsContainer IsContainerTest(int /* dummy */,
0158                             typename C::iterator* /* it */ = NULL,
0159                             typename C::const_iterator* /* const_it */ = NULL) {
0160     return 0;
0161 }
0162 #endif // GTEST_LANG_CXX11
0163
0164 typedef char IsNotContainer;
0165 template <class C>
0166 IsNotContainer IsNotContainerTest(long /* dummy */) { return '\0'; }
0167
0168 // Trait to detect whether a type T is a hash table.
0169 // The heuristic used is that the type contains an inner type `hasher` and does
0170 // not contain an inner type `reverse_iterator`.

```

```
00971 // If the container is iterable in reverse, then order might actually matter.
00972 template <typename T>
00973 struct IsHashTable {
00974     private:
00975         template <typename U>
00976         static char test(typename U::hasher*, typename U::reverse_iterator*);
00977         template <typename U>
00978         static int test(typename U::hasher*, ...);
00979         template <typename U>
00980         static char test(...);
00981     public:
00982         static const bool value = sizeof(test<T>(0, 0)) == sizeof(int);
00983     };
00985
00986 template <typename T>
00987 const bool IsHashTable<T>::value;
00988
00989 template<typename T>
00990 struct VoidT {
00991     typedef void value_type;
00992 };
00993
00994 template <typename T, typename = void>
00995 struct HasValueType : false_type {};
00996 template <typename T>
00997 struct HasValueType<T, VoidT<typename T::value_type> > : true_type {
00998 };
00999
01000 template <typename C,
01001         bool = sizeof(IsContainerTest<C>(0)) == sizeof(IsContainer),
01002         bool = HasValueType<C>::value>
01003 struct IsRecursiveContainerImpl;
01004
01005 template <typename C, bool HV>
01006 struct IsRecursiveContainerImpl<C, false, HV> : public false_type {};
01007
01008 // Since the IsRecursiveContainerImpl depends on the IsContainerTest we need to
01009 // obey the same inconsistencies as the IsContainerTest, namely check if
01010 // something is a container is relying on only const_iterator in C++11 and
01011 // is relying on both const_iterator and iterator otherwise
01012 template <typename C>
01013 struct IsRecursiveContainerImpl<C, true, false> : public false_type {};
01014
01015 template <typename C>
01016 struct IsRecursiveContainerImpl<C, true, true> {
01017     #if GTEST_LANG_CXX11
01018         typedef typename IteratorTraits<typename C::const_iterator>::value_type
01019             value_type;
01020     #else
01021         typedef typename IteratorTraits<typename C::iterator>::value_type value_type;
01022     #endif
01023         typedef is_same<value_type, C> type;
01024 };
01025
01026 // IsRecursiveContainer<Type> is a unary compile-time predicate that
01027 // evaluates whether C is a recursive container type. A recursive container
01028 // type is a container type whose value_type is equal to the container type
01029 // itself. An example for a recursive container type is
01030 // boost::filesystem::path, whose iterator has a value_type that is equal to
01031 // boost::filesystem::path.
01032 template <typename C>
01033 struct IsRecursiveContainer : public IsRecursiveContainerImpl<C>::type {};
01034
01035 // EnableIf<condition>::type is void when 'Cond' is true, and
01036 // undefined when 'Cond' is false. To use SFINAE to make a function
01037 // overload only apply when a particular expression is true, add
01038 // "typename EnableIf<expression>::type* = 0" as the last parameter.
01039 template<bool> struct EnableIf;
01040 template<> struct EnableIf<true> { typedef void type; }; // NOLINT
01041
01042 // Utilities for native arrays.
01043
01044 // ArrayEq() compares two k-dimensional native arrays using the
01045 // elements' operator==, where k can be any integer >= 0. When k is
01046 // 0, ArrayEq() degenerates into comparing a single pair of values.
01047
01048 template <typename T, typename U>
01049 bool ArrayEq(const T* lhs, size_t size, const U* rhs);
01050
01051 // This generic version is used when k is 0.
01052 template <typename T, typename U>
01053 inline bool ArrayEq(const T& lhs, const U& rhs) { return lhs == rhs; }
01054
01055 // This overload is used when k >= 1.
01056 template <typename T, typename U, size_t N>
01057 inline bool ArrayEq(const T(&lhs)[N], const U(&rhs)[N]) {
```

```

01058     return internal::ArrayEq(lhs, N, rhs);
01059 }
01060
01061 // This helper reduces code bloat. If we instead put its logic inside
01062 // the previous ArrayEq() function, arrays with different sizes would
01063 // lead to different copies of the template code.
01064 template <typename T, typename U>
01065 bool ArrayEq(const T* lhs, size_t size, const U* rhs) {
01066     for (size_t i = 0; i != size; i++) {
01067         if (!internal::ArrayEq(lhs[i], rhs[i]))
01068             return false;
01069     }
01070     return true;
01071 }
01072
01073 // Finds the first element in the iterator range [begin, end) that
01074 // equals elem. Element may be a native array type itself.
01075 template <typename Iter, typename Element>
01076 Iter ArrayAwareFind(Iter begin, Iter end, const Element& elem) {
01077     for (Iter it = begin; it != end; ++it) {
01078         if (internal::ArrayEq(*it, elem))
01079             return it;
01080     }
01081     return end;
01082 }
01083
01084 // CopyArray() copies a k-dimensional native array using the elements'
01085 // operator=, where k can be any integer >= 0. When k is 0,
01086 // CopyArray() degenerates into copying a single value.
01087
01088 template <typename T, typename U>
01089 void CopyArray(const T* from, size_t size, U* to);
01090
01091 // This generic version is used when k is 0.
01092 template <typename T, typename U>
01093 inline void CopyArray(const T& from, U* to) { *to = from; }
01094
01095 // This overload is used when k >= 1.
01096 template <typename T, typename U, size_t N>
01097 inline void CopyArray(const T(&from)[N], U(*to)[N]) {
01098     internal::CopyArray(from, N, *to);
01099 }
01100
01101 // This helper reduces code bloat. If we instead put its logic inside
01102 // the previous CopyArray() function, arrays with different sizes
01103 // would lead to different copies of the template code.
01104 template <typename T, typename U>
01105 void CopyArray(const T* from, size_t size, U* to) {
01106     for (size_t i = 0; i != size; i++) {
01107         internal::CopyArray(from[i], to + i);
01108     }
01109 }
01110
01111 // The relation between an NativeArray object (see below) and the
01112 // native array it represents.
01113 // We use 2 different structs to allow non-copyable types to be used, as long
01114 // as RelationToSourceReference() is passed.
01115 struct RelationToSourceReference {};
01116 struct RelationToSourceCopy {};
01117
01118 // Adapts a native array to a read-only STL-style container. Instead
01119 // of the complete STL container concept, this adaptor only implements
01120 // members useful for Google Mock's container matchers. New members
01121 // should be added as needed. To simplify the implementation, we only
01122 // support Element being a raw type (i.e. having no top-level const or
01123 // reference modifier). It's the client's responsibility to satisfy
01124 // this requirement. Element can be an array type itself (hence
01125 // multi-dimensional arrays are supported).
01126 template <typename Element>
01127 class NativeArray {
01128 public:
01129     // STL-style container typedefs.
01130     typedef Element value_type;
01131     typedef Element* iterator;
01132     typedef const Element* const_iterator;
01133
01134     // Constructs from a native array. References the source.
01135     NativeArray(const Element* array, size_t count, RelationToSourceReference) {
01136         InitRef(array, count);
01137     }
01138
01139     // Constructs from a native array. Copies the source.
01140     NativeArray(const Element* array, size_t count, RelationToSourceCopy) {
01141         InitCopy(array, count);
01142     }
01143
01144     // Copy constructor.

```

```

01145     NativeArray(const NativeArray& rhs) {
01146         (this->*rhs.clone_)(rhs.array_, rhs.size_);
01147     }
01148
01149     ~NativeArray() {
01150         if (clone_ != &NativeArray::InitRef)
01151             delete[] array_;
01152     }
01153
01154     // STL-style container methods.
01155     size_t size() const { return size_; }
01156     const_iterator begin() const { return array_; }
01157     const_iterator end() const { return array_ + size_; }
01158     bool operator==(const NativeArray& rhs) const {
01159         return size() == rhs.size() &&
01160             ArrayEq(begin(), size(), rhs.begin());
01161     }
01162
01163 private:
01164     enum {
01165         kCheckTypeIsNotConstOrAReference = StaticAssertTypeEqHelper<
01166             Element, GTEST_REMOVE_REFERENCE_AND_CONST_(Element)>::value
01167     };
01168
01169     // Initializes this object with a copy of the input.
01170     void InitCopy(const Element* array, size_t a_size) {
01171         Element* const copy = new Element[a_size];
01172         CopyArray(array, a_size, copy);
01173         array_ = copy;
01174         size_ = a_size;
01175         clone_ = &NativeArray::InitCopy;
01176     }
01177
01178     // Initializes this object with a reference of the input.
01179     void InitRef(const Element* array, size_t a_size) {
01180         array_ = array;
01181         size_ = a_size;
01182         clone_ = &NativeArray::InitRef;
01183     }
01184
01185     const Element* array_;
01186     size_t size_;
01187     void (NativeArray::*clone_) (const Element*, size_t);
01188
01189     GTEST_DISALLOW_ASSIGN_(NativeArray);
01190 };
01191
01192 } // namespace internal
01193 } // namespace testing
01194
01195 #define GTEST_MESSAGE_AT_(file, line, message, result_type) \
01196     ::testing::internal::AssertHelper(result_type, file, line, message) \
01197     = ::testing::Message()
01198
01199 #define GTEST_MESSAGE_(message, result_type) \
01200     GTEST_MESSAGE_AT_(__FILE__, __LINE__, message, result_type)
01201
01202 #define GTEST_FATAL_FAILURE_(message) \
01203     return GTEST_MESSAGE_(message, ::testing::TestPartResult::kFatalFailure)
01204
01205 #define GTEST_NONFATAL_FAILURE_(message) \
01206     GTEST_MESSAGE_(message, ::testing::TestPartResult::kNonFatalFailure)
01207
01208 #define GTEST_SUCCESS_(message) \
01209     GTEST_MESSAGE_(message, ::testing::TestPartResult::kSuccess)
01210
01211 // Suppress MSVC warning 4702 (unreachable code) for the code following
01212 // statement if it returns or throws (or doesn't return or throw in some
01213 // situations).
01214 #define GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement) \
01215     if (::testing::internal::AlwaysTrue()) { statement; }
01216
01217 #define GTEST_TEST_THROW_(statement, expected_exception, fail) \
01218     GTEST_AMBIGUOUS_ELSE_BLOCKER_ \
01219     if (::testing::internal::ConstCharPtr gtest_msg = "") { \
01220         bool gtest_caught_expected = false; \
01221         try { \
01222             GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement); \
01223         } \
01224         catch (expected_exception const&) { \
01225             gtest_caught_expected = true; \
01226         } \
01227         catch (...) { \
01228             gtest_msg.value = \
01229                 "Expected: " #statement " throws an exception of type " \
01230                 "#expected_exception ".\n Actual: it throws a different type."; \
01231             goto GTEST_CONCAT_TOKEN_(gtest_label_testthrow_, __LINE__); \

```

```

01232     } \
01233     if (!gtest_caught_expected) { \
01234         gtest_msg.value = \
01235             "Expected: " #statement " throws an exception of type " \
01236             "#expected_exception ".\n Actual: it throws nothing."; \
01237         goto GTEST_CONCAT_TOKEN_(gtest_label_testthrow_, __LINE__); \
01238     } \
01239 } else \
01240     GTEST_CONCAT_TOKEN_(gtest_label_testthrow_, __LINE__): \
01241     fail(gtest_msg.value)
01242
01243 #define GTEST_TEST_NO_THROW_(statement, fail) \
01244     GTEST_AMBIGUOUS_ELSE_BLOCKER_ \
01245     if (::testing::internal::AlwaysTrue()) { \
01246         try { \
01247             GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement); \
01248         } \
01249         catch (...) { \
01250             goto GTEST_CONCAT_TOKEN_(gtest_label_testnothrow_, __LINE__); \
01251         } \
01252     } else \
01253         GTEST_CONCAT_TOKEN_(gtest_label_testnothrow_, __LINE__): \
01254             fail("Expected: " #statement " doesn't throw an exception.\n" \
01255                 " Actual: it throws.")
01256
01257 #define GTEST_TEST_ANY_THROW_(statement, fail) \
01258     GTEST_AMBIGUOUS_ELSE_BLOCKER_ \
01259     if (::testing::internal::AlwaysTrue()) { \
01260         bool gtest_caught_any = false; \
01261         try { \
01262             GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement); \
01263         } \
01264         catch (...) { \
01265             gtest_caught_any = true; \
01266         } \
01267         if (!gtest_caught_any) { \
01268             goto GTEST_CONCAT_TOKEN_(gtest_label_testanythrow_, __LINE__); \
01269         } \
01270     } else \
01271         GTEST_CONCAT_TOKEN_(gtest_label_testanythrow_, __LINE__): \
01272             fail("Expected: " #statement " throws an exception.\n" \
01273                 " Actual: it doesn't.")
01274
01275
01276 // Implements Boolean test assertions such as EXPECT_TRUE. expression can be
01277 // either a boolean expression or an AssertionResult. text is a textual
01278 // representation of expression as it was passed into the EXPECT_TRUE.
01279 #define GTEST_TEST_BOOLEAN_(expression, text, actual, expected, fail) \
01280     GTEST_AMBIGUOUS_ELSE_BLOCKER_ \
01281     if (const ::testing::AssertionResult gtest_ar_ = \
01282         ::testing::AssertionResult(expression)) \
01283     ; \
01284     else \
01285         fail(::testing::internal::GetBoolAssertionFailureMessage(\
01286             gtest_ar_, text, #actual, #expected).c_str())
01287
01288 #define GTEST_TEST_NO_FATAL_FAILURE_(statement, fail) \
01289     GTEST_AMBIGUOUS_ELSE_BLOCKER_ \
01290     if (::testing::internal::AlwaysTrue()) { \
01291         ::testing::internal::HasNewFatalFailureHelper gtest_fatal_failure_checker; \
01292         GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement); \
01293         if (gtest_fatal_failure_checker.has_new_fatal_failure()) { \
01294             goto GTEST_CONCAT_TOKEN_(gtest_label_testnofatal_, __LINE__); \
01295         } \
01296     } else \
01297         GTEST_CONCAT_TOKEN_(gtest_label_testnofatal_, __LINE__): \
01298             fail("Expected: " #statement " doesn't generate new fatal " \
01299                 "failures in the current thread.\n" \
01300                 " Actual: it does.")
01301
01302 // Expands to the name of the class that implements the given test.
01303 #define GTEST_TEST_CLASS_NAME_(test_case_name, test_name) \
01304     test_case_name##_##test_name##_Test
01305
01306 // Helper macro for defining tests.
01307 #define GTEST_TEST_(test_case_name, test_name, parent_class, parent_id) \
01308     class GTEST_TEST_CLASS_NAME_(test_case_name, test_name) : public parent_class { \
01309     public: \
01310         GTEST_TEST_CLASS_NAME_(test_case_name, test_name)() {} \
01311     private: \
01312         virtual void TestBody(); \
01313         static ::testing::TestInfo* const test_info_ GTEST_ATTRIBUTE_UNUSED_; \
01314         GTEST_DISALLOW_COPY_AND_ASSIGN_( \
01315             GTEST_TEST_CLASS_NAME_(test_case_name, test_name)); \
01316     }; \
01317 \
01318 ::testing::TestInfo* const GTEST_TEST_CLASS_NAME_(test_case_name, test_name) \

```

```
01319  ::test_info_ =\n01320      ::testing::internal::MakeAndRegisterTestInfo(\n01321          #test_case_name, #test_name, NULL, NULL, \n01322          ::testing::internal::CodeLocation(__FILE__, __LINE__), \n01323          (parent_id), \n01324          parent_class::SetUpTestCase, \n01325          parent_class::TearDownTestCase, \n01326          new ::testing::internal::TestFactoryImpl<\n01327              GTEST_TEST_CLASS_NAME_(test_case_name, test_name)>);\n01328 void GTEST_TEST_CLASS_NAME_(test_case_name, test_name)::TestBody()\n01329\n01330 #endif // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_INTERNAL_H
```

9.39 Dokumentacja pliku **packages/**[**Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-linked_ptr.h**](#)

```
#include <stdlib.h>\n#include <assert.h>\n#include "gtest/internal/gtest-port.h"
```

Komponenty

- class [testing::internal::linked_ptr_internal](#)
- class [testing::internal::linked_ptr< T >](#)

Przestrzenie nazw

- namespace [testing](#)
- namespace [testing::internal](#)

Funkcje

- [GTEST_API_ testing::internal::GTEST_DECLARE_STATIC_MUTEX_ \(g_linked_ptr_mutex\)](#)
- template<typename T>
bool [testing::internal::operator== \(T *ptr, const linked_ptr< T > &x\)](#)
- template<typename T>
bool [testing::internal::operator!= \(T *ptr, const linked_ptr< T > &x\)](#)
- template<typename T>
[linked_ptr< T > testing::internal::make_linked_ptr \(T *ptr\)](#)

9.40 gtest-linked_ptr.h

[Idź do dokumentacji tego pliku.](#)

```
00001 // Copyright 2003 Google Inc.\n00002 // All rights reserved.\n00003 //\n00004 // Redistribution and use in source and binary forms, with or without\n00005 // modification, are permitted provided that the following conditions are\n00006 // met:\n00007 //\n00008 //     * Redistributions of source code must retain the above copyright\n00009 // notice, this list of conditions and the following disclaimer.\n00010 //     * Redistributions in binary form must reproduce the above
```

```
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 // A "smart" pointer type with reference tracking. Every pointer to a
00031 // particular object is kept on a circular linked list. When the last pointer
00032 // to an object is destroyed or reassigned, the object is deleted.
00033 //
00034 // Used properly, this deletes the object when the last reference goes away.
00035 // There are several caveats:
00036 // - Like all reference counting schemes, cycles lead to leaks.
00037 // - Each smart pointer is actually two pointers (8 bytes instead of 4).
00038 // - Every time a pointer is assigned, the entire list of pointers to that
00039 //   object is traversed. This class is therefore NOT SUITABLE when there
00040 //   will often be more than two or three pointers to a particular object.
00041 // - References are only tracked as long as linked_ptr<> objects are copied.
00042 //   If a linked_ptr<> is converted to a raw pointer and back, BAD THINGS
00043 //   will happen (double deletion).
00044 //
00045 // A good use of this class is storing object references in STL containers.
00046 // You can safely put linked_ptr<> in a vector<>.
00047 // Other uses may not be as good.
00048 //
00049 // Note: If you use an incomplete type with linked_ptr<>, the class
00050 // *containing* linked_ptr<> must have a constructor and destructor (even
00051 // if they do nothing!).
00052 //
00053 // Bill Gibbons suggested we use something like this.
00054 //
00055 // Thread Safety:
00056 //   Unlike other linked_ptr implementations, in this implementation
00057 //   a linked_ptr object is thread-safe in the sense that:
00058 //     - it's safe to copy linked_ptr objects concurrently,
00059 //     - it's safe to copy *from* a linked_ptr and read its underlying
00060 //       raw pointer (e.g. via get()) concurrently, and
00061 //     - it's safe to write to two linked_ptrs that point to the same
00062 //       shared object concurrently.
00063 // FIXME: rename this to safe_linked_ptr to avoid
00064 // confusion with normal linked_ptr.
00065
00066 // GOOGLETEST_CM0001 DO NOT DELETE
00067
00068 #ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_LINKED_PTR_H_
00069 #define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_LINKED_PTR_H_
00070
00071 #include <stdlib.h>
00072 #include <assert.h>
00073
00074 #include "gtest/internal/gtest-port.h"
00075
00076 namespace testing {
00077 namespace internal {
00078
00079 // Protects copying of all linked_ptr objects.
00080 GTEST_API_ GTEST_DECLARE_STATIC_MUTEX_(g_linked_ptr_mutex);
00081
00082 // This is used internally by all instances of linked_ptr<>. It needs to be
00083 // a non-template class because different types of linked_ptr<> can refer to
00084 // the same object (linked_ptr<Superclass>(obj) vs linked_ptr<Subclass>(obj)).
00085 // So, it needs to be possible for different types of linked_ptr to participate
00086 // in the same circular linked list, so we need a single class type here.
00087 //
00088 // DO NOT USE THIS CLASS DIRECTLY YOURSELF. Use linked_ptr<T>.
00089 class linked_ptr_internal {
00090 public:
00091     // Create a new circle that includes only this instance.
00092     void join_new() {
00093         next_ = this;
00094     }
00095
00096     // Many linked_ptr operations may change p.link_ for some linked_ptr
00097     // variable p in the same circle as this object. Therefore we need
```

```

00098 // to prevent two such operations from occurring concurrently.
00099 //
00100 // Note that different types of linked_ptr objects can coexist in a
00101 // circle (e.g. linked_ptr<Base>, linked_ptr<Derived1>, and
00102 // linked_ptr<Derived2>). Therefore we must use a single mutex to
00103 // protect all linked_ptr objects. This can create serious
00104 // contention in production code, but is acceptable in a testing
00105 // framework.
00106
00107 // Join an existing circle.
00108 void join(linked_ptr_internal const* ptr)
00109     GTEST_LOCK_EXCLUDED_(g_linked_ptr_mutex) {
00110     MutexLock lock(&g_linked_ptr_mutex);
00111
00112     linked_ptr_internal const* p = ptr;
00113     while (p->next_ != ptr) {
00114         assert(p->next_ != this &&
00115             "Trying to join() a linked ring we are already in. "
00116             "Is GMock thread safety enabled?");
00117         p = p->next_;
00118     }
00119     p->next_ = this;
00120     next_ = ptr;
00121 }
00122
00123 // Leave whatever circle we're part of. Returns true if we were the
00124 // last member of the circle. Once this is done, you can join() another.
00125 bool depart()
00126     GTEST_LOCK_EXCLUDED_(g_linked_ptr_mutex) {
00127     MutexLock lock(&g_linked_ptr_mutex);
00128
00129     if (next_ == this) return true;
00130     linked_ptr_internal const* p = next_;
00131     while (p->next_ != this) {
00132         assert(p->next_ != next_ &&
00133             "Trying to depart() a linked ring we are not in. "
00134             "Is GMock thread safety enabled?");
00135         p = p->next_;
00136     }
00137     p->next_ = next_;
00138     return false;
00139 }
00140
00141 private:
00142     mutable linked_ptr_internal const* next_;
00143 };
00144
00145 template <typename T>
00146 class linked_ptr {
00147 public:
00148     typedef T element_type;
00149
00150     // Take over ownership of a raw pointer. This should happen as soon as
00151     // possible after the object is created.
00152     explicit linked_ptr(T* ptr = NULL) { capture(ptr); }
00153     ~linked_ptr() { depart(); }
00154
00155     // Copy an existing linked_ptr<T>, adding ourselves to the list of references.
00156     template <typename U> linked_ptr(linked_ptr<U> const& ptr) { copy(&ptr); }
00157     linked_ptr(linked_ptr const& ptr) { // NOLINT
00158         assert(&ptr != this);
00159         copy(&ptr);
00160     }
00161
00162     // Assignment releases the old value and acquires the new.
00163     template <typename U> linked_ptr& operator=(linked_ptr<U> const& ptr) {
00164         depart();
00165         copy(&ptr);
00166         return *this;
00167     }
00168
00169     linked_ptr& operator=(linked_ptr const& ptr) {
00170         if (&ptr != this) {
00171             depart();
00172             copy(&ptr);
00173         }
00174         return *this;
00175     }
00176
00177     // Smart pointer members.
00178     void reset(T* ptr = NULL) {
00179         depart();
00180         capture(ptr);
00181     }
00182     T* get() const { return value_; }
00183     T* operator->() const { return value_; }
00184     T& operator*() const { return *value_; }

```

```

00185     bool operator==(T* p) const { return value_ == p; }
00186     bool operator!=(T* p) const { return value_ != p; }
00187     template <typename U>
00188     bool operator==(linked_ptr<U> const& ptr) const {
00189         return value_ == ptr.get();
00190     }
00191     template <typename U>
00192     bool operator!=(linked_ptr<U> const& ptr) const {
00193         return value_ != ptr.get();
00194     }
00195 }
00196
00197 private:
00198     template <typename U>
00199     friend class linked_ptr;
00200
00201     T* value_;
00202     linked_ptr_internal link_;
00203
00204     void depart() {
00205         if (link_.depart()) delete value_;
00206     }
00207
00208     void capture(T* ptr) {
00209         value_ = ptr;
00210         link_.join_new();
00211     }
00212
00213     template <typename U> void copy(linked_ptr<U> const* ptr) {
00214         value_ = ptr->get();
00215         if (value_)
00216             link_.join(&ptr->link_);
00217         else
00218             link_.join_new();
00219     }
00220 };
00221
00222     template<typename T> inline
00223     bool operator==(T* ptr, const linked_ptr<T>& x) {
00224         return ptr == x.get();
00225     }
00226
00227     template<typename T> inline
00228     bool operator!=(T* ptr, const linked_ptr<T>& x) {
00229         return ptr != x.get();
00230     }
00231
00232 // A function to convert T* into linked_ptr<T>
00233 // Doing e.g. make_linked_ptr(new FooBarBaz<type>(arg)) is a shorter notation
00234 // for linked_ptr<FooBarBaz<type> >(new FooBarBaz<type>(arg))
00235     template <typename T>
00236     linked_ptr<T> make_linked_ptr(T* ptr) {
00237         return linked_ptr<T>(ptr);
00238     }
00239
00240 } // namespace internal
00241 } // namespace testing
00242
00243 #endif // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_LINKED_PTR_H_

```

9.41 Dokumentacja pliku

packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-generated.h

```
#include "gtest/internal/gtest-param-util.h"
#include "gtest/internal/gtest-port.h"
```

Komponenty

- class [testing::internal::ValueArray1< T1 >](#)

Przestrzenie nazw

- namespace `testing`
 - namespace `testing::internal`

Funkcje

- template<typename ForwardIterator>
`internal::ParamGenerator< typename ::testing::internal::IteratorTraits< ForwardIterator >::value_type >`
`testing::ValuesIn(ForwardIterator begin, ForwardIterator end)`
- template<typename T, size_t N>
`internal::ParamGenerator< T >` `testing::ValuesIn(const T(&array)[N])`
- template<class Container>
`internal::ParamGenerator< typename Container::value_type >` `testing::ValuesIn(const Container &container)`

9.42 gtest-param-util-generated.h

[Idź do dokumentacji tego pliku.](#)

```

00001 // This file was GENERATED by command:
00002 //     pump.py gtest-param-util-generated.h.pump
00003 // DO NOT EDIT BY HAND!!!
00004
00005 // Copyright 2008 Google Inc.
00006 // All Rights Reserved.
00007 //
00008 // Redistribution and use in source and binary forms, with or without
00009 // modification, are permitted provided that the following conditions are
00010 // met:
00011 //
00012 //      * Redistributions of source code must retain the above copyright
00013 // notice, this list of conditions and the following disclaimer.
00014 //      * Redistributions in binary form must reproduce the above
00015 // copyright notice, this list of conditions and the following disclaimer
00016 // in the documentation and/or other materials provided with the
00017 // distribution.
00018 //      * Neither the name of Google Inc. nor the names of its
00019 // contributors may be used to endorse or promote products derived from
00020 // this software without specific prior written permission.
00021 //
00022 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00023 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00024 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00025 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00026 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00027 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00028 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00029 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00030 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00031 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00032 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00033
00034
00035 // Type and function utilities for implementing parameterized tests.
00036 // This file is generated by a SCRIPT. DO NOT EDIT BY HAND!
00037 //
00038 // Currently Google Test supports at most 50 arguments in Values,
00039 // and at most 10 arguments in Combine. Please contact
00040 // googletestframework@googlegroups.com if you need more.
00041 // Please note that the number of arguments to Combine is limited
00042 // by the maximum arity of the implementation of tuple which is
00043 // currently set at 10.
00044
00045 // GOOGLETEST_CM0001 DO NOT DELETE
00046
00047 #ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PARAM_UTIL_GENERATED_H_
00048 #define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PARAM_UTIL_GENERATED_H_
00049
00050 #include "gtest/internal/gtest-param-util.h"
00051 #include "gtest/internal/gtest-port.h"
00052
00053 namespace testing {
00054
00055 // Forward declarations of ValuesIn(), which is implemented in
00056 // include/gtest/gtest-param-test.h.
00057 template <typename ForwardIterator>
00058 internal::ParamGenerator<
00059     typename ::testing::internal::IteratorTraits<ForwardIterator>::value_type>
00060 ValuesIn(ForwardIterator begin, ForwardIterator end);
00061
00062 template <typename T, size_t N>

```

```
00063 internal::ParamGenerator<T> ValuesIn(const T (&array)[N]);
00064
00065 template <class Container>
00066 internal::ParamGenerator<typename Container::value_type> ValuesIn(
00067     const Container& container);
00068
00069 namespace internal {
00070
00071 // Used in the Values() function to provide polymorphic capabilities.
00072 template <typename T1>
00073 class ValueArray1 {
00074 public:
00075     explicit ValueArray1(T1 v1) : v1_(v1) {}
00076
00077     template <typename T>
00078     operator ParamGenerator<T>() const {
00079         const T array[] = {static_cast<T>(v1_)};
00080         return ValuesIn(array);
00081     }
00082
00083     ValueArray1(const ValueArray1& other) : v1_(other.v1_) {}
00084
00085 private:
00086     // No implementation - assignment is unsupported.
00087     void operator=(const ValueArray1& other);
00088
00089     const T1 v1_;
00090 };
00091
00092 template <typename T1, typename T2>
00093 class ValueArray2 {
00094 public:
00095     ValueArray2(T1 v1, T2 v2) : v1_(v1), v2_(v2) {}
00096
00097     template <typename T>
00098     operator ParamGenerator<T>() const {
00099         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_)};
00100         return ValuesIn(array);
00101     }
00102
00103     ValueArray2(const ValueArray2& other) : v1_(other.v1_), v2_(other.v2_) {}
00104
00105 private:
00106     // No implementation - assignment is unsupported.
00107     void operator=(const ValueArray2& other);
00108
00109     const T1 v1_;
00110     const T2 v2_;
00111 };
00112
00113 template <typename T1, typename T2, typename T3>
00114 class ValueArray3 {
00115 public:
00116     ValueArray3(T1 v1, T2 v2, T3 v3) : v1_(v1), v2_(v2), v3_(v3) {}
00117
00118     template <typename T>
00119     operator ParamGenerator<T>() const {
00120         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00121                         static_cast<T>(v3_)};
00122         return ValuesIn(array);
00123     }
00124
00125     ValueArray3(const ValueArray3& other) : v1_(other.v1_), v2_(other.v2_),
00126         v3_(other.v3_) {}
00127
00128 private:
00129     // No implementation - assignment is unsupported.
00130     void operator=(const ValueArray3& other);
00131
00132     const T1 v1_;
00133     const T2 v2_;
00134     const T3 v3_;
00135 };
00136
00137 template <typename T1, typename T2, typename T3, typename T4>
00138 class ValueArray4 {
00139 public:
00140     ValueArray4(T1 v1, T2 v2, T3 v3, T4 v4) : v1_(v1), v2_(v2), v3_(v3),
00141         v4_(v4) {}
00142
00143     template <typename T>
00144     operator ParamGenerator<T>() const {
00145         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00146                         static_cast<T>(v3_), static_cast<T>(v4_)};
00147         return ValuesIn(array);
00148     }
00149 }
```

```

00150     ValueArray4(const ValueArray4& other) : v1_(other.v1_), v2_(other.v2_),
00151         v3_(other.v3_), v4_(other.v4_) {}
00152
00153 private:
00154     // No implementation - assignment is unsupported.
00155     void operator=(const ValueArray4& other);
00156
00157     const T1 v1_;
00158     const T2 v2_;
00159     const T3 v3_;
00160     const T4 v4_;
00161 };
00162
00163 template <typename T1, typename T2, typename T3, typename T4, typename T5>
00164 class ValueArray5 {
00165     public:
00166     ValueArray5(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5) : v1_(v1), v2_(v2), v3_(v3),
00167         v4_(v4), v5_(v5) {}
00168
00169     template <typename T>
00170     operator ParamGenerator<T>() const {
00171         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00172             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_)};
00173         return ValuesIn(array);
00174     }
00175
00176     ValueArray5(const ValueArray5& other) : v1_(other.v1_), v2_(other.v2_),
00177         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_) {}
00178
00179     private:
00180     // No implementation - assignment is unsupported.
00181     void operator=(const ValueArray5& other);
00182
00183     const T1 v1_;
00184     const T2 v2_;
00185     const T3 v3_;
00186     const T4 v4_;
00187     const T5 v5_;
00188 };
00189
00190 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00191     typename T6>
00192 class ValueArray6 {
00193     public:
00194     ValueArray6(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6) : v1_(v1), v2_(v2),
00195         v3_(v3), v4_(v4), v5_(v5), v6_(v6) {}
00196
00197     template <typename T>
00198     operator ParamGenerator<T>() const {
00199         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00200             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00201             static_cast<T>(v6_)};
00202         return ValuesIn(array);
00203     }
00204
00205     ValueArray6(const ValueArray6& other) : v1_(other.v1_), v2_(other.v2_),
00206         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_) {}
00207
00208     private:
00209     // No implementation - assignment is unsupported.
00210     void operator=(const ValueArray6& other);
00211
00212     const T1 v1_;
00213     const T2 v2_;
00214     const T3 v3_;
00215     const T4 v4_;
00216     const T5 v5_;
00217     const T6 v6_;
00218 };
00219
00220 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00221     typename T6, typename T7>
00222 class ValueArray7 {
00223     public:
00224     ValueArray7(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7) : v1_(v1),
00225         v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7) {}
00226
00227     template <typename T>
00228     operator ParamGenerator<T>() const {
00229         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00230             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00231             static_cast<T>(v6_), static_cast<T>(v7_)};
00232         return ValuesIn(array);
00233     }
00234
00235     ValueArray7(const ValueArray7& other) : v1_(other.v1_), v2_(other.v2_),
00236         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),

```

```

00237     v7_(other.v7_) {}
00238
00239 private:
00240 // No implementation - assignment is unsupported.
00241 void operator=(const ValueArray7& other);
00242
00243 const T1 v1_;
00244 const T2 v2_;
00245 const T3 v3_;
00246 const T4 v4_;
00247 const T5 v5_;
00248 const T6 v6_;
00249 const T7 v7_;
00250 };
00251
00252 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00253   typename T6, typename T7, typename T8>
00254 class ValueArray8 {
00255 public:
00256     ValueArray8(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7,
00257                 T8 v8) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
00258                 v8_(v8) {}
00259
00260     template <typename T>
00261     operator ParamGenerator<T>() const {
00262         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00263                             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00264                             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_)};
00265         return ValuesIn(array);
00266     }
00267
00268     ValueArray8(const ValueArray8& other) : v1_(other.v1_), v2_(other.v2_),
00269                 v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
00270                 v7_(other.v7_), v8_(other.v8_) {}
00271
00272 private:
00273 // No implementation - assignment is unsupported.
00274 void operator=(const ValueArray8& other);
00275
00276 const T1 v1_;
00277 const T2 v2_;
00278 const T3 v3_;
00279 const T4 v4_;
00280 const T5 v5_;
00281 const T6 v6_;
00282 const T7 v7_;
00283 const T8 v8_;
00284 };
00285
00286 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00287   typename T6, typename T7, typename T8, typename T9>
00288 class ValueArray9 {
00289 public:
00290     ValueArray9(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8,
00291                 T9 v9) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
00292                 v8_(v8), v9_(v9) {}
00293
00294     template <typename T>
00295     operator ParamGenerator<T>() const {
00296         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00297                             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00298                             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
00299                             static_cast<T>(v9_)};
00300         return ValuesIn(array);
00301     }
00302
00303     ValueArray9(const ValueArray9& other) : v1_(other.v1_), v2_(other.v2_),
00304                 v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
00305                 v7_(other.v7_), v8_(other.v8_), v9_(other.v9_) {}
00306
00307 private:
00308 // No implementation - assignment is unsupported.
00309 void operator=(const ValueArray9& other);
00310
00311 const T1 v1_;
00312 const T2 v2_;
00313 const T3 v3_;
00314 const T4 v4_;
00315 const T5 v5_;
00316 const T6 v6_;
00317 const T7 v7_;
00318 const T8 v8_;
00319 const T9 v9_;
00320 };
00321
00322 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00323   typename T6, typename T7, typename T8, typename T9, typename T10>

```

```

00324 class ValueArray10 {
00325 public:
00326     ValueArray10(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00327                 T10 v10) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
00328                 v8_(v8), v9_(v9), v10_(v10) {}
00329
00330     template <typename T>
00331     operator ParamGenerator<T>() const {
00332         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00333                         static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00334                         static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
00335                         static_cast<T>(v9_), static_cast<T>(v10_)};
00336         return ValuesIn(array);
00337     }
00338
00339     ValueArray10(const ValueArray10& other) : v1_(other.v1_), v2_(other.v2_),
00340             v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
00341             v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_) {}
00342
00343 private:
00344     // No implementation - assignment is unsupported.
00345     void operator=(const ValueArray10& other);
00346
00347     const T1 v1_;
00348     const T2 v2_;
00349     const T3 v3_;
00350     const T4 v4_;
00351     const T5 v5_;
00352     const T6 v6_;
00353     const T7 v7_;
00354     const T8 v8_;
00355     const T9 v9_;
00356     const T10 v10_;
00357 };
00358
00359     template <typename T1, typename T2, typename T3, typename T4, typename T5,
00360             typename T6, typename T7, typename T8, typename T9, typename T10,
00361             typename T11>
00362     class ValueArray11 {
00363 public:
00364     ValueArray11(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00365                 T10 v10, T11 v11) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6),
00366                 v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11) {}
00367
00368     template <typename T>
00369     operator ParamGenerator<T>() const {
00370         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00371                         static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00372                         static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
00373                         static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_)};
00374         return ValuesIn(array);
00375     }
00376
00377     ValueArray11(const ValueArray11& other) : v1_(other.v1_), v2_(other.v2_),
00378             v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
00379             v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
00380             v11_(other.v11_) {}
00381
00382 private:
00383     // No implementation - assignment is unsupported.
00384     void operator=(const ValueArray11& other);
00385
00386     const T1 v1_;
00387     const T2 v2_;
00388     const T3 v3_;
00389     const T4 v4_;
00390     const T5 v5_;
00391     const T6 v6_;
00392     const T7 v7_;
00393     const T8 v8_;
00394     const T9 v9_;
00395     const T10 v10_;
00396     const T11 v11_;
00397 };
00398
00399     template <typename T1, typename T2, typename T3, typename T4, typename T5,
00400             typename T6, typename T7, typename T8, typename T9, typename T10,
00401             typename T11, typename T12>
00402     class ValueArray12 {
00403 public:
00404     ValueArray12(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00405                 T10 v10, T11 v11, T12 v12) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5),
00406                 v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12) {}
00407
00408     template <typename T>
00409     operator ParamGenerator<T>() const {
00410         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),

```

```

00411     static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00412     static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
00413     static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
00414     static_cast<T>(v12_));
00415     return ValuesIn(array);
00416 }
00417
00418 ValueArray12(const ValueArray12& other) : v1_(other.v1_), v2_(other.v2_),
00419     v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
00420     v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
00421     v11_(other.v11_), v12_(other.v12_) {}
00422
00423 private:
00424 // No implementation - assignment is unsupported.
00425 void operator=(const ValueArray12& other);
00426
00427 const T1 v1_;
00428 const T2 v2_;
00429 const T3 v3_;
00430 const T4 v4_;
00431 const T5 v5_;
00432 const T6 v6_;
00433 const T7 v7_;
00434 const T8 v8_;
00435 const T9 v9_;
00436 const T10 v10_;
00437 const T11 v11_;
00438 const T12 v12_;
00439 };
00440
00441 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00442     typename T6, typename T7, typename T8, typename T9, typename T10,
00443     typename T11, typename T12, typename T13>
00444 class ValueArray13 {
00445 public:
00446     ValueArray13(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00447         T10 v10, T11 v11, T12 v12, T13 v13) : v1_(v1), v2_(v2), v3_(v3), v4_(v4),
00448         v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11),
00449         v12_(v12), v13_(v13) {}
00450
00451     template <typename T>
00452     operator ParamGenerator<T>() const {
00453         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00454             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00455             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
00456             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
00457             static_cast<T>(v12_), static_cast<T>(v13_)};
00458         return ValuesIn(array);
00459     }
00460
00461     ValueArray13(const ValueArray13& other) : v1_(other.v1_), v2_(other.v2_),
00462         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
00463         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
00464         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_) {}
00465
00466 private:
00467 // No implementation - assignment is unsupported.
00468 void operator=(const ValueArray13& other);
00469
00470 const T1 v1_;
00471 const T2 v2_;
00472 const T3 v3_;
00473 const T4 v4_;
00474 const T5 v5_;
00475 const T6 v6_;
00476 const T7 v7_;
00477 const T8 v8_;
00478 const T9 v9_;
00479 const T10 v10_;
00480 const T11 v11_;
00481 const T12 v12_;
00482 const T13 v13_;
00483 };
00484
00485 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00486     typename T6, typename T7, typename T8, typename T9, typename T10,
00487     typename T11, typename T12, typename T13, typename T14>
00488 class ValueArray14 {
00489 public:
00490     ValueArray14(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00491         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14) : v1_(v1), v2_(v2), v3_(v3),
00492         v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
00493         v11_(v11), v12_(v12), v13_(v13), v14_(v14) {}
00494
00495     template <typename T>
00496     operator ParamGenerator<T>() const {
00497         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),

```

```

00498     static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00499     static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
00500     static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
00501     static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_)};
00502     return ValuesIn(array);
00503 }
00504
00505 ValueArray14(const ValueArray14& other) : v1_(other.v1_), v2_(other.v2_),
00506     v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
00507     v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
00508     v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_) {}
00509
00510 private:
00511 // No implementation - assignment is unsupported.
00512 void operator=(const ValueArray14& other);
00513
00514 const T1 v1_;
00515 const T2 v2_;
00516 const T3 v3_;
00517 const T4 v4_;
00518 const T5 v5_;
00519 const T6 v6_;
00520 const T7 v7_;
00521 const T8 v8_;
00522 const T9 v9_;
00523 const T10 v10_;
00524 const T11 v11_;
00525 const T12 v12_;
00526 const T13 v13_;
00527 const T14 v14_;
00528 };
00529
00530 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00531     typename T6, typename T7, typename T8, typename T9, typename T10,
00532     typename T11, typename T12, typename T13, typename T14, typename T15>
00533 class ValueArray15 {
00534 public:
00535     ValueArray15(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00536         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15) : v1_(v1), v2_(v2),
00537         v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
00538         v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15) {}
00539
00540 template <typename T>
00541 operator ParamGenerator<T>() const {
00542     const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00543         static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00544         static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
00545         static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
00546         static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
00547         static_cast<T>(v15_)};
00548     return ValuesIn(array);
00549 }
00550
00551 ValueArray15(const ValueArray15& other) : v1_(other.v1_), v2_(other.v2_),
00552     v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
00553     v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
00554     v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
00555     v15_(other.v15_) {}
00556
00557 private:
00558 // No implementation - assignment is unsupported.
00559 void operator=(const ValueArray15& other);
00560
00561 const T1 v1_;
00562 const T2 v2_;
00563 const T3 v3_;
00564 const T4 v4_;
00565 const T5 v5_;
00566 const T6 v6_;
00567 const T7 v7_;
00568 const T8 v8_;
00569 const T9 v9_;
00570 const T10 v10_;
00571 const T11 v11_;
00572 const T12 v12_;
00573 const T13 v13_;
00574 const T14 v14_;
00575 const T15 v15_;
00576 };
00577
00578 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00579     typename T6, typename T7, typename T8, typename T9, typename T10,
00580     typename T11, typename T12, typename T13, typename T14, typename T15,
00581     typename T16>
00582 class ValueArray16 {
00583 public:
00584     ValueArray16(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,

```

```

00585     T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16) : v1_(v1),
00586     v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9),
00587     v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15),
00588     v16_(v16) {}
00589
00590     template <typename T>
00591     operator ParamGenerator<T>() const {
00592         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00593             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00594             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
00595             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
00596             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
00597             static_cast<T>(v15_), static_cast<T>(v16_)};
00598         return ValuesIn(array);
00599     }
00600
00601     ValueArray16(const ValueArray16& other) : v1_(other.v1_), v2_(other.v2_),
00602         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
00603         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
00604         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
00605         v15_(other.v15_), v16_(other.v16_) {}
00606
00607     private:
00608     // No implementation - assignment is unsupported.
00609     void operator=(const ValueArray16& other);
00610
00611     const T1 v1_;
00612     const T2 v2_;
00613     const T3 v3_;
00614     const T4 v4_;
00615     const T5 v5_;
00616     const T6 v6_;
00617     const T7 v7_;
00618     const T8 v8_;
00619     const T9 v9_;
00620     const T10 v10_;
00621     const T11 v11_;
00622     const T12 v12_;
00623     const T13 v13_;
00624     const T14 v14_;
00625     const T15 v15_;
00626     const T16 v16_;
00627 };
00628
00629     template <typename T1, typename T2, typename T3, typename T4, typename T5,
00630         typename T6, typename T7, typename T8, typename T9, typename T10,
00631         typename T11, typename T12, typename T13, typename T14, typename T15,
00632         typename T16, typename T17>
00633     class ValueArray17 {
00634     public:
00635         ValueArray17(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00636             T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16,
00637             T17 v17) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
00638             v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
00639             v15_(v15), v16_(v16), v17_(v17) {}
00640
00641     template <typename T>
00642     operator ParamGenerator<T>() const {
00643         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00644             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00645             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
00646             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
00647             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
00648             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_)};
00649         return ValuesIn(array);
00650     }
00651
00652     ValueArray17(const ValueArray17& other) : v1_(other.v1_), v2_(other.v2_),
00653         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
00654         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
00655         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
00656         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_) {}
00657
00658     private:
00659     // No implementation - assignment is unsupported.
00660     void operator=(const ValueArray17& other);
00661
00662     const T1 v1_;
00663     const T2 v2_;
00664     const T3 v3_;
00665     const T4 v4_;
00666     const T5 v5_;
00667     const T6 v6_;
00668     const T7 v7_;
00669     const T8 v8_;
00670     const T9 v9_;
00671     const T10 v10_;

```

```

00672     const T11 v11_;
00673     const T12 v12_;
00674     const T13 v13_;
00675     const T14 v14_;
00676     const T15 v15_;
00677     const T16 v16_;
00678     const T17 v17_;
00679 };
00680
00681 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00682     typename T6, typename T7, typename T8, typename T9, typename T10,
00683     typename T11, typename T12, typename T13, typename T14, typename T15,
00684     typename T16, typename T17, typename T18>
00685 class ValueArray18 {
00686 public:
00687     ValueArray18(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00688         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
00689         T18 v18) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
00690         v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
00691         v15_(v15), v16_(v16), v17_(v17), v18_(v18) {}
00692
00693     template <typename T>
00694     operator ParamGenerator<T>() const {
00695         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00696             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00697             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
00698             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
00699             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
00700             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
00701             static_cast<T>(v18_)};
00702         return ValuesIn(array);
00703     }
00704
00705     ValueArray18(const ValueArray18& other) : v1_(other.v1_), v2_(other.v2_),
00706         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
00707         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
00708         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
00709         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_) {}
00710
00711 private:
00712 // No implementation - assignment is unsupported.
00713 void operator=(const ValueArray18& other);
00714
00715     const T1 v1_;
00716     const T2 v2_;
00717     const T3 v3_;
00718     const T4 v4_;
00719     const T5 v5_;
00720     const T6 v6_;
00721     const T7 v7_;
00722     const T8 v8_;
00723     const T9 v9_;
00724     const T10 v10_;
00725     const T11 v11_;
00726     const T12 v12_;
00727     const T13 v13_;
00728     const T14 v14_;
00729     const T15 v15_;
00730     const T16 v16_;
00731     const T17 v17_;
00732     const T18 v18_;
00733 };
00734
00735 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00736     typename T6, typename T7, typename T8, typename T9, typename T10,
00737     typename T11, typename T12, typename T13, typename T14, typename T15,
00738     typename T16, typename T17, typename T18, typename T19>
00739 class ValueArray19 {
00740 public:
00741     ValueArray19(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00742         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
00743         T18 v18, T19 v19) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6),
00744         v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13),
00745         v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19) {}
00746
00747     template <typename T>
00748     operator ParamGenerator<T>() const {
00749         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00750             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00751             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
00752             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
00753             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
00754             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
00755             static_cast<T>(v18_), static_cast<T>(v19_)};
00756         return ValuesIn(array);
00757     }
00758 }
```

```

00759     ValueArray19(const ValueArray19& other) : v1_(other.v1_), v2_(other.v2_),
00760         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
00761         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
00762         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
00763         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
00764         v19_(other.v19_) {}
00765
00766     private:
00767     // No implementation - assignment is unsupported.
00768     void operator=(const ValueArray19& other);
00769
00770     const T1 v1_;
00771     const T2 v2_;
00772     const T3 v3_;
00773     const T4 v4_;
00774     const T5 v5_;
00775     const T6 v6_;
00776     const T7 v7_;
00777     const T8 v8_;
00778     const T9 v9_;
00779     const T10 v10_;
00780     const T11 v11_;
00781     const T12 v12_;
00782     const T13 v13_;
00783     const T14 v14_;
00784     const T15 v15_;
00785     const T16 v16_;
00786     const T17 v17_;
00787     const T18 v18_;
00788     const T19 v19_;
00789 }
00790
00791 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00792     typename T6, typename T7, typename T8, typename T9, typename T10,
00793     typename T11, typename T12, typename T13, typename T14, typename T15,
00794     typename T16, typename T17, typename T18, typename T19, typename T20>
00795 class ValueArray20 {
00796     public:
00797     ValueArray20(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00798             T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
00799             T18 v18, T19 v19, T20 v20) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5),
00800             v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12),
00801             v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18),
00802             v19_(v19), v20_(v20) {}
00803
00804     template <typename T>
00805     operator ParamGenerator<T>() const {
00806         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00807             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00808             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
00809             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
00810             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
00811             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
00812             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_)};
00813         return ValuesIn(array);
00814     }
00815
00816     ValueArray20(const ValueArray20& other) : v1_(other.v1_), v2_(other.v2_),
00817         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
00818         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
00819         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
00820         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
00821         v19_(other.v19_), v20_(other.v20_) {}
00822
00823     private:
00824     // No implementation - assignment is unsupported.
00825     void operator=(const ValueArray20& other);
00826
00827     const T1 v1_;
00828     const T2 v2_;
00829     const T3 v3_;
00830     const T4 v4_;
00831     const T5 v5_;
00832     const T6 v6_;
00833     const T7 v7_;
00834     const T8 v8_;
00835     const T9 v9_;
00836     const T10 v10_;
00837     const T11 v11_;
00838     const T12 v12_;
00839     const T13 v13_;
00840     const T14 v14_;
00841     const T15 v15_;
00842     const T16 v16_;
00843     const T17 v17_;
00844     const T18 v18_;
00845     const T19 v19_;

```

```

00846 const T20 v20_;
00847 };
00848
00849 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00850     typename T6, typename T7, typename T8, typename T9, typename T10,
00851     typename T11, typename T12, typename T13, typename T14, typename T15,
00852     typename T16, typename T17, typename T18, typename T19, typename T20,
00853     typename T21>
00854 class ValueArray21 {
00855 public:
00856     ValueArray21(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00857                 T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
00858                 T18 v18, T19 v19, T20 v20, T21 v21) : v1_(v1), v2_(v2), v3_(v3), v4_(v4),
00859                 v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11),
00860                 v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17),
00861                 v18_(v18), v19_(v19), v20_(v20), v21_(v21) {}
00862
00863 template <typename T>
00864 operator ParamGenerator<T>() const {
00865     const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00866                         static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00867                         static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
00868                         static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
00869                         static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
00870                         static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
00871                         static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
00872                         static_cast<T>(v21_)};
00873     return ValuesIn(array);
00874 }
00875
00876 ValueArray21(const ValueArray21& other) : v1_(other.v1_), v2_(other.v2_),
00877                 v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
00878                 v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
00879                 v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
00880                 v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
00881                 v19_(other.v19_), v20_(other.v20_), v21_(other.v21_) {}
00882
00883 private:
00884 // No implementation - assignment is unsupported.
00885 void operator=(const ValueArray21& other);
00886
00887 const T1 v1_;
00888 const T2 v2_;
00889 const T3 v3_;
00890 const T4 v4_;
00891 const T5 v5_;
00892 const T6 v6_;
00893 const T7 v7_;
00894 const T8 v8_;
00895 const T9 v9_;
00896 const T10 v10_;
00897 const T11 v11_;
00898 const T12 v12_;
00899 const T13 v13_;
00900 const T14 v14_;
00901 const T15 v15_;
00902 const T16 v16_;
00903 const T17 v17_;
00904 const T18 v18_;
00905 const T19 v19_;
00906 const T20 v20_;
00907 const T21 v21_;
00908 };
00909
00910 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00911     typename T6, typename T7, typename T8, typename T9, typename T10,
00912     typename T11, typename T12, typename T13, typename T14, typename T15,
00913     typename T16, typename T17, typename T18, typename T19, typename T20,
00914     typename T21, typename T22>
00915 class ValueArray22 {
00916 public:
00917     ValueArray22(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00918                 T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
00919                 T18 v18, T19 v19, T20 v20, T21 v21, T22 v22) : v1_(v1), v2_(v2), v3_(v3),
00920                 v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
00921                 v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
00922                 v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22) {}
00923
00924 template <typename T>
00925 operator ParamGenerator<T>() const {
00926     const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00927                         static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00928                         static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
00929                         static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
00930                         static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
00931                         static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
00932                         static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_)};

```

```

00933     static_cast<T>(v21_), static_cast<T>(v22_) };
00934     return ValuesIn(array);
00935 }
00936
00937 ValueArray22(const ValueArray22& other) : v1_(other.v1_), v2_(other.v2_),
00938     v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
00939     v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
00940     v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
00941     v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
00942     v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_) {}
00943
00944 private:
00945 // No implementation - assignment is unsupported.
00946 void operator=(const ValueArray22& other);
00947
00948 const T1 v1_;
00949 const T2 v2_;
00950 const T3 v3_;
00951 const T4 v4_;
00952 const T5 v5_;
00953 const T6 v6_;
00954 const T7 v7_;
00955 const T8 v8_;
00956 const T9 v9_;
00957 const T10 v10_;
00958 const T11 v11_;
00959 const T12 v12_;
00960 const T13 v13_;
00961 const T14 v14_;
00962 const T15 v15_;
00963 const T16 v16_;
00964 const T17 v17_;
00965 const T18 v18_;
00966 const T19 v19_;
00967 const T20 v20_;
00968 const T21 v21_;
00969 const T22 v22_;
00970 };
00971
00972 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00973     typename T6, typename T7, typename T8, typename T9, typename T10,
00974     typename T11, typename T12, typename T13, typename T14, typename T15,
00975     typename T16, typename T17, typename T18, typename T19, typename T20,
00976     typename T21, typename T22, typename T23>
00977 class ValueArray23 {
00978 public:
00979     ValueArray23(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
00980                 T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
00981                 T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23) : v1_(v1), v2_(v2),
00982                 v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
00983                 v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
00984                 v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22),
00985                 v23_(v23) {}
00986
00987 template <typename T>
00988 operator ParamGenerator<T>() const {
00989     const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
00990                         static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
00991                         static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
00992                         static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
00993                         static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
00994                         static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
00995                         static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
00996                         static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_)};
00997     return ValuesIn(array);
00998 }
00999
01000 ValueArray23(const ValueArray23& other) : v1_(other.v1_), v2_(other.v2_),
01001     v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
01002     v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
01003     v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
01004     v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
01005     v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
01006     v23_(other.v23_) {}
01007
01008 private:
01009 // No implementation - assignment is unsupported.
01010 void operator=(const ValueArray23& other);
01011
01012 const T1 v1_;
01013 const T2 v2_;
01014 const T3 v3_;
01015 const T4 v4_;
01016 const T5 v5_;
01017 const T6 v6_;
01018 const T7 v7_;
01019 const T8 v8_;

```

```

01020 const T9 v9_;
01021 const T10 v10_;
01022 const T11 v11_;
01023 const T12 v12_;
01024 const T13 v13_;
01025 const T14 v14_;
01026 const T15 v15_;
01027 const T16 v16_;
01028 const T17 v17_;
01029 const T18 v18_;
01030 const T19 v19_;
01031 const T20 v20_;
01032 const T21 v21_;
01033 const T22 v22_;
01034 const T23 v23_;
01035 };
01036
01037 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01038     typename T6, typename T7, typename T8, typename T9, typename T10,
01039     typename T11, typename T12, typename T13, typename T14, typename T15,
01040     typename T16, typename T17, typename T18, typename T19, typename T20,
01041     typename T21, typename T22, typename T23, typename T24>
01042 class ValueArray24 {
01043 public:
01044     ValueArray24(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
01045         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
01046         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24) : v1_(v1),
01047         v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9),
01048         v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15),
01049         v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21),
01050         v22_(v22), v23_(v23), v24_(v24) {}
01051
01052     template <typename T>
01053     operator ParamGenerator<T>() const {
01054         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
01055             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
01056             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
01057             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
01058             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
01059             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
01060             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
01061             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
01062             static_cast<T>(v24_)};
01063         return ValuesIn(array);
01064     }
01065
01066     ValueArray24(const ValueArray24& other) : v1_(other.v1_), v2_(other.v2_),
01067         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
01068         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
01069         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
01070         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
01071         v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
01072         v23_(other.v23_), v24_(other.v24_) {}
01073
01074 private:
01075     // No implementation - assignment is unsupported.
01076     void operator=(const ValueArray24& other);
01077
01078     const T1 v1_;
01079     const T2 v2_;
01080     const T3 v3_;
01081     const T4 v4_;
01082     const T5 v5_;
01083     const T6 v6_;
01084     const T7 v7_;
01085     const T8 v8_;
01086     const T9 v9_;
01087     const T10 v10_;
01088     const T11 v11_;
01089     const T12 v12_;
01090     const T13 v13_;
01091     const T14 v14_;
01092     const T15 v15_;
01093     const T16 v16_;
01094     const T17 v17_;
01095     const T18 v18_;
01096     const T19 v19_;
01097     const T20 v20_;
01098     const T21 v21_;
01099     const T22 v22_;
01100    const T23 v23_;
01101    const T24 v24_;
01102 };
01103
01104 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01105     typename T6, typename T7, typename T8, typename T9, typename T10,
01106     typename T11, typename T12, typename T13, typename T14, typename T15,

```

```

01107     typename T16, typename T17, typename T18, typename T19, typename T20,
01108     typename T21, typename T22, typename T23, typename T24, typename T25>
01109 class ValueArray25 {
01110 public:
01111     ValueArray25(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
01112         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
01113         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24,
01114         T25 v25) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
01115         v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
01116         v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
01117         v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25) {}
01118
01119     template <typename T>
01120     operator ParamGenerator<T>() const {
01121         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
01122             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
01123             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
01124             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
01125             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
01126             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
01127             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
01128             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
01129             static_cast<T>(v24_), static_cast<T>(v25_)};
01130         return ValuesIn(array);
01131     }
01132
01133     ValueArray25(const ValueArray25& other) : v1_(other.v1_), v2_(other.v2_),
01134         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
01135         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
01136         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
01137         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
01138         v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
01139         v23_(other.v23_), v24_(other.v24_), v25_(other.v25_) {}
01140
01141 private:
01142     // No implementation - assignment is unsupported.
01143     void operator=(const ValueArray25& other);
01144
01145     const T1 v1_;
01146     const T2 v2_;
01147     const T3 v3_;
01148     const T4 v4_;
01149     const T5 v5_;
01150     const T6 v6_;
01151     const T7 v7_;
01152     const T8 v8_;
01153     const T9 v9_;
01154     const T10 v10_;
01155     const T11 v11_;
01156     const T12 v12_;
01157     const T13 v13_;
01158     const T14 v14_;
01159     const T15 v15_;
01160     const T16 v16_;
01161     const T17 v17_;
01162     const T18 v18_;
01163     const T19 v19_;
01164     const T20 v20_;
01165     const T21 v21_;
01166     const T22 v22_;
01167     const T23 v23_;
01168     const T24 v24_;
01169     const T25 v25_;
01170 };
01171
01172     template <typename T1, typename T2, typename T3, typename T4, typename T5,
01173         typename T6, typename T7, typename T8, typename T9, typename T10,
01174         typename T11, typename T12, typename T13, typename T14, typename T15,
01175         typename T16, typename T17, typename T18, typename T19, typename T20,
01176         typename T21, typename T22, typename T23, typename T24, typename T25,
01177         typename T26>
01178 class ValueArray26 {
01179 public:
01180     ValueArray26(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
01181         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
01182         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
01183         T26 v26) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
01184         v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
01185         v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
01186         v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26) {}
01187
01188     template <typename T>
01189     operator ParamGenerator<T>() const {
01190         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
01191             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
01192             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
01193             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_)},

```

```

01194     static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
01195     static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
01196     static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
01197     static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
01198     static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_)};
01199   return ValuesIn(array);
01200 }
01201
01202 ValueArray26(const ValueArray26& other) : v1_(other.v1_), v2_(other.v2_),
01203   v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
01204   v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
01205   v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
01206   v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
01207   v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
01208   v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_) {}
01209
01210 private:
01211 // No implementation - assignment is unsupported.
01212 void operator=(const ValueArray26& other);
01213
01214 const T1 v1_;
01215 const T2 v2_;
01216 const T3 v3_;
01217 const T4 v4_;
01218 const T5 v5_;
01219 const T6 v6_;
01220 const T7 v7_;
01221 const T8 v8_;
01222 const T9 v9_;
01223 const T10 v10_;
01224 const T11 v11_;
01225 const T12 v12_;
01226 const T13 v13_;
01227 const T14 v14_;
01228 const T15 v15_;
01229 const T16 v16_;
01230 const T17 v17_;
01231 const T18 v18_;
01232 const T19 v19_;
01233 const T20 v20_;
01234 const T21 v21_;
01235 const T22 v22_;
01236 const T23 v23_;
01237 const T24 v24_;
01238 const T25 v25_;
01239 const T26 v26_;
01240 };
01241
01242 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01243   typename T6, typename T7, typename T8, typename T9, typename T10,
01244   typename T11, typename T12, typename T13, typename T14, typename T15,
01245   typename T16, typename T17, typename T18, typename T19, typename T20,
01246   typename T21, typename T22, typename T23, typename T24, typename T25,
01247   typename T26, typename T27>
01248 class ValueArray27 {
01249 public:
01250   ValueArray27(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
01251     T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
01252     T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
01253     T26 v26, T27 v27) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6),
01254     v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13),
01255     v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19),
01256     v20_(v20), v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25),
01257     v26_(v26), v27_(v27) {}
01258
01259 template <typename T>
01260 operator ParamGenerator<T>() const {
01261   const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
01262     static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
01263     static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
01264     static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
01265     static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
01266     static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
01267     static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
01268     static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
01269     static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
01270     static_cast<T>(v27_)};
01271   return ValuesIn(array);
01272 }
01273
01274 ValueArray27(const ValueArray27& other) : v1_(other.v1_), v2_(other.v2_),
01275   v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
01276   v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
01277   v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
01278   v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
01279   v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
01280   v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
```

```

01281     v27_(other.v27_) {}
01282
01283 private:
01284 // No implementation - assignment is unsupported.
01285 void operator=(const ValueArray27& other);
01286
01287 const T1 v1_;
01288 const T2 v2_;
01289 const T3 v3_;
01290 const T4 v4_;
01291 const T5 v5_;
01292 const T6 v6_;
01293 const T7 v7_;
01294 const T8 v8_;
01295 const T9 v9_;
01296 const T10 v10_;
01297 const T11 v11_;
01298 const T12 v12_;
01299 const T13 v13_;
01300 const T14 v14_;
01301 const T15 v15_;
01302 const T16 v16_;
01303 const T17 v17_;
01304 const T18 v18_;
01305 const T19 v19_;
01306 const T20 v20_;
01307 const T21 v21_;
01308 const T22 v22_;
01309 const T23 v23_;
01310 const T24 v24_;
01311 const T25 v25_;
01312 const T26 v26_;
01313 const T27 v27_;
01314 };
01315
01316 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01317     typename T6, typename T7, typename T8, typename T9, typename T10,
01318     typename T11, typename T12, typename T13, typename T14, typename T15,
01319     typename T16, typename T17, typename T18, typename T19, typename T20,
01320     typename T21, typename T22, typename T23, typename T24, typename T25,
01321     typename T26, typename T27, typename T28>
01322 class ValueArray28 {
01323 public:
01324     ValueArray28(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
01325         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
01326         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
01327         T26 v26, T27 v27, T28 v28) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5),
01328         v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12),
01329         v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18),
01330         v19_(v19), v20_(v20), v21_(v21), v22_(v22), v23_(v23), v24_(v24),
01331         v25_(v25), v26_(v26), v27_(v27), v28_(v28) {}
01332
01333 template <typename T>
01334 operator ParamGenerator<T>() const {
01335     const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
01336         static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
01337         static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
01338         static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
01339         static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
01340         static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
01341         static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
01342         static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
01343         static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
01344         static_cast<T>(v27_), static_cast<T>(v28_)};
01345     return ValuesIn(array);
01346 }
01347
01348 ValueArray28(const ValueArray28& other) : v1_(other.v1_), v2_(other.v2_),
01349     v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
01350     v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
01351     v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
01352     v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
01353     v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
01354     v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
01355     v27_(other.v27_), v28_(other.v28_) {}
01356
01357 private:
01358 // No implementation - assignment is unsupported.
01359 void operator=(const ValueArray28& other);
01360
01361 const T1 v1_;
01362 const T2 v2_;
01363 const T3 v3_;
01364 const T4 v4_;
01365 const T5 v5_;
01366 const T6 v6_;
01367 const T7 v7_;

```

```

01368 const T8 v8_;
01369 const T9 v9_;
01370 const T10 v10_;
01371 const T11 v11_;
01372 const T12 v12_;
01373 const T13 v13_;
01374 const T14 v14_;
01375 const T15 v15_;
01376 const T16 v16_;
01377 const T17 v17_;
01378 const T18 v18_;
01379 const T19 v19_;
01380 const T20 v20_;
01381 const T21 v21_;
01382 const T22 v22_;
01383 const T23 v23_;
01384 const T24 v24_;
01385 const T25 v25_;
01386 const T26 v26_;
01387 const T27 v27_;
01388 const T28 v28_;
01389 };
01390
01391 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01392     typename T6, typename T7, typename T8, typename T9, typename T10,
01393     typename T11, typename T12, typename T13, typename T14, typename T15,
01394     typename T16, typename T17, typename T18, typename T19, typename T20,
01395     typename T21, typename T22, typename T23, typename T24, typename T25,
01396     typename T26, typename T27, typename T28, typename T29>
01397 class ValueArray29 {
01398 public:
01399     ValueArray29(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
01400         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
01401         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
01402         T26 v26, T27 v27, T28 v28, T29 v29) : v1_(v1), v2_(v2), v3_(v3), v4_(v4),
01403         v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11),
01404         v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17),
01405         v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22), v23_(v23),
01406         v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28), v29_(v29) {}
01407
01408     template <typename T>
01409     operator ParamGenerator<T>() const {
01410         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
01411             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
01412             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
01413             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
01414             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
01415             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
01416             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
01417             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
01418             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
01419             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_)};
01420         return ValuesIn(array);
01421     }
01422
01423     ValueArray29(const ValueArray29& other) : v1_(other.v1_), v2_(other.v2_),
01424         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
01425         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
01426         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
01427         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
01428         v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
01429         v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
01430         v27_(other.v27_), v28_(other.v28_), v29_(other.v29_) {}
01431
01432 private:
01433     // No implementation - assignment is unsupported.
01434     void operator=(const ValueArray29& other);
01435
01436     const T1 v1_;
01437     const T2 v2_;
01438     const T3 v3_;
01439     const T4 v4_;
01440     const T5 v5_;
01441     const T6 v6_;
01442     const T7 v7_;
01443     const T8 v8_;
01444     const T9 v9_;
01445     const T10 v10_;
01446     const T11 v11_;
01447     const T12 v12_;
01448     const T13 v13_;
01449     const T14 v14_;
01450     const T15 v15_;
01451     const T16 v16_;
01452     const T17 v17_;
01453     const T18 v18_;
01454     const T19 v19_;

```

```

01455     const T20 v20_;
01456     const T21 v21_;
01457     const T22 v22_;
01458     const T23 v23_;
01459     const T24 v24_;
01460     const T25 v25_;
01461     const T26 v26_;
01462     const T27 v27_;
01463     const T28 v28_;
01464     const T29 v29_;
01465 };
01466
01467 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01468     typename T6, typename T7, typename T8, typename T9, typename T10,
01469     typename T11, typename T12, typename T13, typename T14, typename T15,
01470     typename T16, typename T17, typename T18, typename T19, typename T20,
01471     typename T21, typename T22, typename T23, typename T24, typename T25,
01472     typename T26, typename T27, typename T28, typename T29, typename T30>
01473 class ValueArray30 {
01474 public:
01475     ValueArray30(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
01476         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
01477         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
01478         T26 v26, T27 v27, T28 v28, T29 v29, T30 v30) : v1_(v1), v2_(v2), v3_(v3),
01479         v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
01480         v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
01481         v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22),
01482         v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28),
01483         v29_(v29), v30_(v30) {}
01484
01485     template <typename T>
01486     operator ParamGenerator<T>() const {
01487         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
01488             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
01489             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
01490             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
01491             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
01492             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
01493             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
01494             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
01495             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
01496             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
01497             static_cast<T>(v30_)};
01498         return ValuesIn(array);
01499     }
01500
01501     ValueArray30(const ValueArray30& other) : v1_(other.v1_), v2_(other.v2_),
01502         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
01503         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
01504         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
01505         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
01506         v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
01507         v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
01508         v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_) {}
01509
01510 private:
01511     // No implementation - assignment is unsupported.
01512     void operator=(const ValueArray30& other);
01513
01514     const T1 v1_;
01515     const T2 v2_;
01516     const T3 v3_;
01517     const T4 v4_;
01518     const T5 v5_;
01519     const T6 v6_;
01520     const T7 v7_;
01521     const T8 v8_;
01522     const T9 v9_;
01523     const T10 v10_;
01524     const T11 v11_;
01525     const T12 v12_;
01526     const T13 v13_;
01527     const T14 v14_;
01528     const T15 v15_;
01529     const T16 v16_;
01530     const T17 v17_;
01531     const T18 v18_;
01532     const T19 v19_;
01533     const T20 v20_;
01534     const T21 v21_;
01535     const T22 v22_;
01536     const T23 v23_;
01537     const T24 v24_;
01538     const T25 v25_;
01539     const T26 v26_;
01540     const T27 v27_;
01541     const T28 v28_;

```

```

01542     const T29 v29_;
01543     const T30 v30_;
01544 };
01545
01546 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01547     typename T6, typename T7, typename T8, typename T9, typename T10,
01548     typename T11, typename T12, typename T13, typename T14, typename T15,
01549     typename T16, typename T17, typename T18, typename T19, typename T20,
01550     typename T21, typename T22, typename T23, typename T24, typename T25,
01551     typename T26, typename T27, typename T28, typename T29, typename T30,
01552     typename T31>
01553 class ValueArray31 {
01554 public:
01555     ValueArray31(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
01556         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
01557         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
01558         T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31) : v1_(v1), v2_(v2),
01559         v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
01560         v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
01561         v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22),
01562         v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28),
01563         v29_(v29), v30_(v30), v31_(v31) {}
01564
01565     template <typename T>
01566     operator ParamGenerator<T>() const {
01567         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
01568             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
01569             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
01570             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
01571             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
01572             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
01573             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
01574             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
01575             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
01576             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
01577             static_cast<T>(v30_), static_cast<T>(v31_)};
01578         return ValuesIn(array);
01579     }
01580
01581     ValueArray31(const ValueArray31& other) : v1_(other.v1_), v2_(other.v2_),
01582         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
01583         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
01584         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
01585         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
01586         v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
01587         v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
01588         v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
01589         v31_(other.v31_) {}
01590
01591 private:
01592     // No implementation - assignment is unsupported.
01593     void operator=(const ValueArray31& other);
01594
01595     const T1 v1_;
01596     const T2 v2_;
01597     const T3 v3_;
01598     const T4 v4_;
01599     const T5 v5_;
01600     const T6 v6_;
01601     const T7 v7_;
01602     const T8 v8_;
01603     const T9 v9_;
01604     const T10 v10_;
01605     const T11 v11_;
01606     const T12 v12_;
01607     const T13 v13_;
01608     const T14 v14_;
01609     const T15 v15_;
01610     const T16 v16_;
01611     const T17 v17_;
01612     const T18 v18_;
01613     const T19 v19_;
01614     const T20 v20_;
01615     const T21 v21_;
01616     const T22 v22_;
01617     const T23 v23_;
01618     const T24 v24_;
01619     const T25 v25_;
01620     const T26 v26_;
01621     const T27 v27_;
01622     const T28 v28_;
01623     const T29 v29_;
01624     const T30 v30_;
01625     const T31 v31_;
01626 };
01627
01628 template <typename T1, typename T2, typename T3, typename T4, typename T5,

```

```

01629     typename T6, typename T7, typename T8, typename T9, typename T10,
01630     typename T11, typename T12, typename T13, typename T14, typename T15,
01631     typename T16, typename T17, typename T18, typename T19, typename T20,
01632     typename T21, typename T22, typename T23, typename T24, typename T25,
01633     typename T26, typename T27, typename T28, typename T29, typename T30,
01634     typename T31, typename T32>
01635 class ValueArray32 {
01636 public:
01637     ValueArray32(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
01638                 T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
01639                 T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
01640                 T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32) : v1_(v1),
01641                 v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9),
01642                 v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15),
01643                 v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21),
01644                 v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27),
01645                 v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32) {}
01646
01647     template <typename T>
01648     operator ParamGenerator<T>() const {
01649         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
01650                     static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
01651                     static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
01652                     static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
01653                     static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
01654                     static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
01655                     static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
01656                     static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
01657                     static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
01658                     static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
01659                     static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_)};
01660         return ValuesIn(array);
01661     }
01662
01663     ValueArray32(const ValueArray32& other) : v1_(other.v1_), v2_(other.v2_),
01664                 v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
01665                 v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
01666                 v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
01667                 v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
01668                 v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
01669                 v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
01670                 v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
01671                 v31_(other.v31_), v32_(other.v32_) {}
01672
01673 private:
01674     // No implementation - assignment is unsupported.
01675     void operator=(const ValueArray32& other);
01676
01677     const T1 v1_;
01678     const T2 v2_;
01679     const T3 v3_;
01680     const T4 v4_;
01681     const T5 v5_;
01682     const T6 v6_;
01683     const T7 v7_;
01684     const T8 v8_;
01685     const T9 v9_;
01686     const T10 v10_;
01687     const T11 v11_;
01688     const T12 v12_;
01689     const T13 v13_;
01690     const T14 v14_;
01691     const T15 v15_;
01692     const T16 v16_;
01693     const T17 v17_;
01694     const T18 v18_;
01695     const T19 v19_;
01696     const T20 v20_;
01697     const T21 v21_;
01698     const T22 v22_;
01699     const T23 v23_;
01700     const T24 v24_;
01701     const T25 v25_;
01702     const T26 v26_;
01703     const T27 v27_;
01704     const T28 v28_;
01705     const T29 v29_;
01706     const T30 v30_;
01707     const T31 v31_;
01708     const T32 v32_;
01709 };
01710
01711 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01712         typename T6, typename T7, typename T8, typename T9, typename T10,
01713         typename T11, typename T12, typename T13, typename T14, typename T15,
01714         typename T16, typename T17, typename T18, typename T19, typename T20,
01715         typename T21, typename T22, typename T23, typename T24, typename T25,
```

```

01716     typename T26, typename T27, typename T28, typename T29, typename T30,
01717     typename T31, typename T32, typename T33>
01718 class ValueArray33 {
01719 public:
01720     ValueArray33(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
01721             T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
01722             T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
01723             T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32,
01724             T33 v33) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
01725             v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
01726             v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
01727             v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26),
01728             v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32),
01729             v33_(v33) {}
01730
01731     template <typename T>
01732     operator ParamGenerator<T>() const {
01733         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
01734             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
01735             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
01736             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
01737             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
01738             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
01739             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
01740             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
01741             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
01742             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
01743             static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
01744             static_cast<T>(v33_)};
01745         return ValuesIn(array);
01746     }
01747
01748     ValueArray33(const ValueArray33& other) : v1_(other.v1_), v2_(other.v2_),
01749             v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
01750             v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
01751             v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
01752             v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
01753             v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
01754             v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
01755             v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
01756             v31_(other.v31_), v32_(other.v32_), v33_(other.v33_) {}
01757
01758 private:
01759     // No implementation - assignment is unsupported.
01760     void operator=(const ValueArray33& other);
01761
01762     const T1 v1_;
01763     const T2 v2_;
01764     const T3 v3_;
01765     const T4 v4_;
01766     const T5 v5_;
01767     const T6 v6_;
01768     const T7 v7_;
01769     const T8 v8_;
01770     const T9 v9_;
01771     const T10 v10_;
01772     const T11 v11_;
01773     const T12 v12_;
01774     const T13 v13_;
01775     const T14 v14_;
01776     const T15 v15_;
01777     const T16 v16_;
01778     const T17 v17_;
01779     const T18 v18_;
01780     const T19 v19_;
01781     const T20 v20_;
01782     const T21 v21_;
01783     const T22 v22_;
01784     const T23 v23_;
01785     const T24 v24_;
01786     const T25 v25_;
01787     const T26 v26_;
01788     const T27 v27_;
01789     const T28 v28_;
01790     const T29 v29_;
01791     const T30 v30_;
01792     const T31 v31_;
01793     const T32 v32_;
01794     const T33 v33_;
01795 };
01796
01797 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01798     typename T6, typename T7, typename T8, typename T9, typename T10,
01799     typename T11, typename T12, typename T13, typename T14, typename T15,
01800     typename T16, typename T17, typename T18, typename T19, typename T20,
01801     typename T21, typename T22, typename T23, typename T24, typename T25,
01802     typename T26, typename T27, typename T28, typename T29, typename T30,

```

```

01803     typename T31, typename T32, typename T33, typename T34>
01804 class ValueArray34 {
01805 public:
01806     ValueArray34(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
01807                 T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
01808                 T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
01809                 T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
01810                 T34 v34) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
01811                 v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
01812                 v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
01813                 v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26),
01814                 v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32),
01815                 v33_(v33), v34_(v34) {}

01816
01817     template <typename T>
01818     operator ParamGenerator<T>() const {
01819         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
01820                             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
01821                             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
01822                             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
01823                             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
01824                             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
01825                             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
01826                             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
01827                             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
01828                             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
01829                             static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
01830                             static_cast<T>(v33_), static_cast<T>(v34_)};
01831         return ValuesIn(array);
01832     }
01833
01834     ValueArray34(const ValueArray34& other) : v1_(other.v1_), v2_(other.v2_),
01835                 v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
01836                 v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
01837                 v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
01838                 v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
01839                 v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
01840                 v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
01841                 v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
01842                 v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_) {}
01843
01844 private:
01845     // No implementation - assignment is unsupported.
01846     void operator=(const ValueArray34& other);
01847
01848     const T1 v1_;
01849     const T2 v2_;
01850     const T3 v3_;
01851     const T4 v4_;
01852     const T5 v5_;
01853     const T6 v6_;
01854     const T7 v7_;
01855     const T8 v8_;
01856     const T9 v9_;
01857     const T10 v10_;
01858     const T11 v11_;
01859     const T12 v12_;
01860     const T13 v13_;
01861     const T14 v14_;
01862     const T15 v15_;
01863     const T16 v16_;
01864     const T17 v17_;
01865     const T18 v18_;
01866     const T19 v19_;
01867     const T20 v20_;
01868     const T21 v21_;
01869     const T22 v22_;
01870     const T23 v23_;
01871     const T24 v24_;
01872     const T25 v25_;
01873     const T26 v26_;
01874     const T27 v27_;
01875     const T28 v28_;
01876     const T29 v29_;
01877     const T30 v30_;
01878     const T31 v31_;
01879     const T32 v32_;
01880     const T33 v33_;
01881     const T34 v34_;
01882 };
01883
01884 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01885             typename T6, typename T7, typename T8, typename T9, typename T10,
01886             typename T11, typename T12, typename T13, typename T14, typename T15,
01887             typename T16, typename T17, typename T18, typename T19, typename T20,
01888             typename T21, typename T22, typename T23, typename T24, typename T25,
01889             typename T26, typename T27, typename T28, typename T29, typename T30,
```

```

01890     typename T31, typename T32, typename T33, typename T34, typename T35>
01891 class ValueArray35 {
01892 public:
01893     ValueArray35(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
01894                 T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
01895                 T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
01896                 T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
01897                 T34 v34, T35 v35) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6),
01898                 v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13),
01899                 v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19),
01900                 v20_(v20), v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25),
01901                 v26_(v26), v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31),
01902                 v32_(v32), v33_(v33), v34_(v34), v35_(v35) {}
01903
01904     template <typename T>
01905     operator ParamGenerator<T>() const {
01906         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
01907                             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
01908                             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
01909                             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
01910                             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
01911                             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
01912                             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
01913                             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
01914                             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
01915                             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
01916                             static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
01917                             static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_) };
01918         return ValuesIn(array);
01919     }
01920
01921     ValueArray35(const ValueArray35& other) : v1_(other.v1_), v2_(other.v2_),
01922                 v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
01923                 v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
01924                 v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
01925                 v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
01926                 v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
01927                 v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
01928                 v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
01929                 v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
01930                 v35_(other.v35_) {}
01931
01932 private:
01933     // No implementation - assignment is unsupported.
01934     void operator=(const ValueArray35& other);
01935
01936     const T1 v1_;
01937     const T2 v2_;
01938     const T3 v3_;
01939     const T4 v4_;
01940     const T5 v5_;
01941     const T6 v6_;
01942     const T7 v7_;
01943     const T8 v8_;
01944     const T9 v9_;
01945     const T10 v10_;
01946     const T11 v11_;
01947     const T12 v12_;
01948     const T13 v13_;
01949     const T14 v14_;
01950     const T15 v15_;
01951     const T16 v16_;
01952     const T17 v17_;
01953     const T18 v18_;
01954     const T19 v19_;
01955     const T20 v20_;
01956     const T21 v21_;
01957     const T22 v22_;
01958     const T23 v23_;
01959     const T24 v24_;
01960     const T25 v25_;
01961     const T26 v26_;
01962     const T27 v27_;
01963     const T28 v28_;
01964     const T29 v29_;
01965     const T30 v30_;
01966     const T31 v31_;
01967     const T32 v32_;
01968     const T33 v33_;
01969     const T34 v34_;
01970     const T35 v35_;
01971 };
01972
01973 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01974             typename T6, typename T7, typename T8, typename T9, typename T10,
01975             typename T11, typename T12, typename T13, typename T14, typename T15,
01976             typename T16, typename T17, typename T18, typename T19, typename T20,

```

```

01977     typename T21, typename T22, typename T23, typename T24, typename T25,
01978     typename T26, typename T27, typename T28, typename T29, typename T30,
01979     typename T31, typename T32, typename T33, typename T34, typename T35,
01980     typename T36>
01981 class ValueArray36 {
01982 public:
01983     ValueArray36(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
01984         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
01985         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
01986         T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
01987         T34 v34, T35 v35, T36 v36) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5),
01988         v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12),
01989         v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18),
01990         v19_(v19), v20_(v20), v21_(v21), v22_(v22), v23_(v23), v24_(v24),
01991         v25_(v25), v26_(v26), v27_(v27), v28_(v28), v29_(v29), v30_(v30),
01992         v31_(v31), v32_(v32), v33_(v33), v34_(v34), v35_(v35), v36_(v36) {}
01993
01994     template <typename T>
01995     operator ParamGenerator<T>() const {
01996         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
01997             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
01998             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
01999             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
02000             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
02001             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
02002             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
02003             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
02004             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
02005             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
02006             static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
02007             static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
02008             static_cast<T>(v36_)};
02009         return ValuesIn(array);
02010     }
02011
02012     ValueArray36(const ValueArray36& other) : v1_(other.v1_), v2_(other.v2_),
02013         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
02014         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
02015         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
02016         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
02017         v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
02018         v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
02019         v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
02020         v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
02021         v35_(other.v35_), v36_(other.v36_) {}
02022
02023 private:
02024     // No implementation - assignment is unsupported.
02025     void operator=(const ValueArray36& other);
02026
02027     const T1 v1_;
02028     const T2 v2_;
02029     const T3 v3_;
02030     const T4 v4_;
02031     const T5 v5_;
02032     const T6 v6_;
02033     const T7 v7_;
02034     const T8 v8_;
02035     const T9 v9_;
02036     const T10 v10_;
02037     const T11 v11_;
02038     const T12 v12_;
02039     const T13 v13_;
02040     const T14 v14_;
02041     const T15 v15_;
02042     const T16 v16_;
02043     const T17 v17_;
02044     const T18 v18_;
02045     const T19 v19_;
02046     const T20 v20_;
02047     const T21 v21_;
02048     const T22 v22_;
02049     const T23 v23_;
02050     const T24 v24_;
02051     const T25 v25_;
02052     const T26 v26_;
02053     const T27 v27_;
02054     const T28 v28_;
02055     const T29 v29_;
02056     const T30 v30_;
02057     const T31 v31_;
02058     const T32 v32_;
02059     const T33 v33_;
02060     const T34 v34_;
02061     const T35 v35_;
02062     const T36 v36_;
02063 };

```

```

02064
02065 template <typename T1, typename T2, typename T3, typename T4, typename T5,
02066     typename T6, typename T7, typename T8, typename T9, typename T10,
02067     typename T11, typename T12, typename T13, typename T14, typename T15,
02068     typename T16, typename T17, typename T18, typename T19, typename T20,
02069     typename T21, typename T22, typename T23, typename T24, typename T25,
02070     typename T26, typename T27, typename T28, typename T29, typename T30,
02071     typename T31, typename T32, typename T33, typename T34, typename T35,
02072     typename T36, typename T37>
02073 class ValueArray37 {
02074 public:
02075     ValueArray37(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
02076         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
02077         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
02078         T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
02079         T34 v34, T35 v35, T36 v36, T37 v37) : v1_(v1), v2_(v2), v3_(v3), v4_(v4),
02080         v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11),
02081         v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17),
02082         v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22), v23_(v23),
02083         v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28), v29_(v29),
02084         v30_(v30), v31_(v31), v32_(v32), v33_(v33), v34_(v34), v35_(v35),
02085         v36_(v36), v37_(v37) {}
02086
02087     template <typename T>
02088     operator ParamGenerator<T>() const {
02089         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
02090             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
02091             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
02092             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
02093             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
02094             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
02095             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
02096             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
02097             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
02098             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
02099             static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
02100             static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
02101             static_cast<T>(v36_), static_cast<T>(v37_)};
02102         return ValuesIn(array);
02103     }
02104
02105     ValueArray37(const ValueArray37& other) : v1_(other.v1_), v2_(other.v2_),
02106         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
02107         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
02108         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
02109         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
02110         v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
02111         v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
02112         v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
02113         v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
02114         v35_(other.v35_), v36_(other.v36_), v37_(other.v37_) {}
02115
02116     private:
02117     // No implementation - assignment is unsupported.
02118     void operator=(const ValueArray37& other);
02119
02120     const T1 v1_;
02121     const T2 v2_;
02122     const T3 v3_;
02123     const T4 v4_;
02124     const T5 v5_;
02125     const T6 v6_;
02126     const T7 v7_;
02127     const T8 v8_;
02128     const T9 v9_;
02129     const T10 v10_;
02130     const T11 v11_;
02131     const T12 v12_;
02132     const T13 v13_;
02133     const T14 v14_;
02134     const T15 v15_;
02135     const T16 v16_;
02136     const T17 v17_;
02137     const T18 v18_;
02138     const T19 v19_;
02139     const T20 v20_;
02140     const T21 v21_;
02141     const T22 v22_;
02142     const T23 v23_;
02143     const T24 v24_;
02144     const T25 v25_;
02145     const T26 v26_;
02146     const T27 v27_;
02147     const T28 v28_;
02148     const T29 v29_;
02149     const T30 v30_;
02150     const T31 v31_;

```

```

02151 const T32 v32_;
02152 const T33 v33_;
02153 const T34 v34_;
02154 const T35 v35_;
02155 const T36 v36_;
02156 const T37 v37_;
02157 };
02158
02159 template <typename T1, typename T2, typename T3, typename T4, typename T5,
02160     typename T6, typename T7, typename T8, typename T9, typename T10,
02161     typename T11, typename T12, typename T13, typename T14, typename T15,
02162     typename T16, typename T17, typename T18, typename T19, typename T20,
02163     typename T21, typename T22, typename T23, typename T24, typename T25,
02164     typename T26, typename T27, typename T28, typename T29, typename T30,
02165     typename T31, typename T32, typename T33, typename T34, typename T35,
02166     typename T36, typename T37, typename T38>
02167 class ValueArray38 {
02168 public:
02169     ValueArray38(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
02170         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
02171         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
02172         T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
02173         T34 v34, T35 v35, T36 v36, T37 v37, T38 v38) : v1_(v1), v2_(v2), v3_(v3),
02174         v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
02175         v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
02176         v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22),
02177         v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28),
02178         v29_(v29), v30_(v30), v31_(v31), v32_(v32), v33_(v33), v34_(v34),
02179         v35_(v35), v36_(v36), v37_(v37), v38_(v38) {}
02180
02181     template <typename T>
02182     operator ParamGenerator<T>() const {
02183         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
02184             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
02185             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
02186             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
02187             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
02188             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
02189             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
02190             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
02191             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
02192             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
02193             static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
02194             static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
02195             static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_)};
02196         return ValuesIn(array);
02197     }
02198
02199     ValueArray38(const ValueArray38& other) : v1_(other.v1_), v2_(other.v2_),
02200         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
02201         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
02202         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
02203         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
02204         v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
02205         v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
02206         v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
02207         v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
02208         v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_) {}
02209
02210     private:
02211     // No implementation - assignment is unsupported.
02212     void operator=(const ValueArray38& other);
02213
02214     const T1 v1_;
02215     const T2 v2_;
02216     const T3 v3_;
02217     const T4 v4_;
02218     const T5 v5_;
02219     const T6 v6_;
02220     const T7 v7_;
02221     const T8 v8_;
02222     const T9 v9_;
02223     const T10 v10_;
02224     const T11 v11_;
02225     const T12 v12_;
02226     const T13 v13_;
02227     const T14 v14_;
02228     const T15 v15_;
02229     const T16 v16_;
02230     const T17 v17_;
02231     const T18 v18_;
02232     const T19 v19_;
02233     const T20 v20_;
02234     const T21 v21_;
02235     const T22 v22_;
02236     const T23 v23_;
02237     const T24 v24_;

```

```

02238 const T25 v25_;
02239 const T26 v26_;
02240 const T27 v27_;
02241 const T28 v28_;
02242 const T29 v29_;
02243 const T30 v30_;
02244 const T31 v31_;
02245 const T32 v32_;
02246 const T33 v33_;
02247 const T34 v34_;
02248 const T35 v35_;
02249 const T36 v36_;
02250 const T37 v37_;
02251 const T38 v38_;
02252 };
02253
02254 template <typename T1, typename T2, typename T3, typename T4, typename T5,
02255     typename T6, typename T7, typename T8, typename T9, typename T10,
02256     typename T11, typename T12, typename T13, typename T14, typename T15,
02257     typename T16, typename T17, typename T18, typename T19, typename T20,
02258     typename T21, typename T22, typename T23, typename T24, typename T25,
02259     typename T26, typename T27, typename T28, typename T29, typename T30,
02260     typename T31, typename T32, typename T33, typename T34, typename T35,
02261     typename T36, typename T37, typename T38, typename T39>
02262 class ValueArray39 {
02263 public:
02264     ValueArray39(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
02265         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
02266         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
02267         T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
02268         T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39) : v1_(v1), v2_(v2),
02269         v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
02270         v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
02271         v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22),
02272         v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28),
02273         v29_(v29), v30_(v30), v31_(v31), v32_(v32), v33_(v33), v34_(v34),
02274         v35_(v35), v36_(v36), v37_(v37), v38_(v38), v39_(v39) {}
02275
02276     template <typename T>
02277     operator ParamGenerator<T>() const {
02278         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
02279             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
02280             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
02281             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
02282             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
02283             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
02284             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
02285             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
02286             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
02287             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
02288             static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
02289             static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
02290             static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
02291             static_cast<T>(v39_)};
02292         return ValuesIn(array);
02293     }
02294
02295     ValueArray39(const ValueArray39& other) : v1_(other.v1_), v2_(other.v2_),
02296         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
02297         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
02298         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
02299         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
02300         v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
02301         v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
02302         v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
02303         v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
02304         v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
02305         v39_(other.v39_) {}
02306
02307     private:
02308     // No implementation - assignment is unsupported.
02309     void operator=(const ValueArray39& other);
02310
02311     const T1 v1_;
02312     const T2 v2_;
02313     const T3 v3_;
02314     const T4 v4_;
02315     const T5 v5_;
02316     const T6 v6_;
02317     const T7 v7_;
02318     const T8 v8_;
02319     const T9 v9_;
02320     const T10 v10_;
02321     const T11 v11_;
02322     const T12 v12_;
02323     const T13 v13_;
02324     const T14 v14_;

```

```

02325     const T15 v15_;
02326     const T16 v16_;
02327     const T17 v17_;
02328     const T18 v18_;
02329     const T19 v19_;
02330     const T20 v20_;
02331     const T21 v21_;
02332     const T22 v22_;
02333     const T23 v23_;
02334     const T24 v24_;
02335     const T25 v25_;
02336     const T26 v26_;
02337     const T27 v27_;
02338     const T28 v28_;
02339     const T29 v29_;
02340     const T30 v30_;
02341     const T31 v31_;
02342     const T32 v32_;
02343     const T33 v33_;
02344     const T34 v34_;
02345     const T35 v35_;
02346     const T36 v36_;
02347     const T37 v37_;
02348     const T38 v38_;
02349     const T39 v39_;
02350 };
02351
02352 template <typename T1, typename T2, typename T3, typename T4, typename T5,
02353     typename T6, typename T7, typename T8, typename T9, typename T10,
02354     typename T11, typename T12, typename T13, typename T14, typename T15,
02355     typename T16, typename T17, typename T18, typename T19, typename T20,
02356     typename T21, typename T22, typename T23, typename T24, typename T25,
02357     typename T26, typename T27, typename T28, typename T29, typename T30,
02358     typename T31, typename T32, typename T33, typename T34, typename T35,
02359     typename T36, typename T37, typename T38, typename T39, typename T40>
02360 class ValueArray40 {
02361 public:
02362     ValueArray40(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
02363         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
02364         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
02365         T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
02366         T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40) : v1_(v1),
02367         v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9),
02368         v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15),
02369         v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21),
02370         v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27),
02371         v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32), v33_(v33),
02372         v34_(v34), v35_(v35), v36_(v36), v37_(v37), v38_(v38), v39_(v39),
02373         v40_(v40) {}
02374
02375     template <typename T>
02376     operator ParamGenerator<T>() const {
02377         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
02378             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
02379             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
02380             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
02381             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
02382             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
02383             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
02384             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
02385             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
02386             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
02387             static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
02388             static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
02389             static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
02390             static_cast<T>(v39_), static_cast<T>(v40_)};
02391         return ValuesIn(array);
02392     }
02393
02394     ValueArray40(const ValueArray40& other) : v1_(other.v1_), v2_(other.v2_),
02395         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
02396         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
02397         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
02398         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
02399         v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
02400         v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
02401         v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
02402         v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
02403         v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
02404         v39_(other.v39_), v40_(other.v40_) {}
02405
02406     private:
02407     // No implementation - assignment is unsupported.
02408     void operator=(const ValueArray40& other);
02409
02410     const T1 v1_;
02411     const T2 v2_;

```

```

02412 const T3 v3_;
02413 const T4 v4_;
02414 const T5 v5_;
02415 const T6 v6_;
02416 const T7 v7_;
02417 const T8 v8_;
02418 const T9 v9_;
02419 const T10 v10_;
02420 const T11 v11_;
02421 const T12 v12_;
02422 const T13 v13_;
02423 const T14 v14_;
02424 const T15 v15_;
02425 const T16 v16_;
02426 const T17 v17_;
02427 const T18 v18_;
02428 const T19 v19_;
02429 const T20 v20_;
02430 const T21 v21_;
02431 const T22 v22_;
02432 const T23 v23_;
02433 const T24 v24_;
02434 const T25 v25_;
02435 const T26 v26_;
02436 const T27 v27_;
02437 const T28 v28_;
02438 const T29 v29_;
02439 const T30 v30_;
02440 const T31 v31_;
02441 const T32 v32_;
02442 const T33 v33_;
02443 const T34 v34_;
02444 const T35 v35_;
02445 const T36 v36_;
02446 const T37 v37_;
02447 const T38 v38_;
02448 const T39 v39_;
02449 const T40 v40_;
02450 };
02451
02452 template <typename T1, typename T2, typename T3, typename T4, typename T5,
02453     typename T6, typename T7, typename T8, typename T9, typename T10,
02454     typename T11, typename T12, typename T13, typename T14, typename T15,
02455     typename T16, typename T17, typename T18, typename T19, typename T20,
02456     typename T21, typename T22, typename T23, typename T24, typename T25,
02457     typename T26, typename T27, typename T28, typename T29, typename T30,
02458     typename T31, typename T32, typename T33, typename T34, typename T35,
02459     typename T36, typename T37, typename T38, typename T39, typename T40,
02460     typename T41>
02461 class ValueArray41 {
02462 public:
02463     ValueArray41(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
02464         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
02465         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
02466         T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
02467         T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40,
02468         T41 v41) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
02469         v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
02470         v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
02471         v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26),
02472         v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32),
02473         v33_(v33), v34_(v34), v35_(v35), v36_(v36), v37_(v37), v38_(v38),
02474         v39_(v39), v40_(v40), v41_(v41) {}
02475
02476     template <typename T>
02477     operator ParamGenerator<T>() const {
02478         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
02479             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
02480             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
02481             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
02482             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
02483             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
02484             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
02485             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
02486             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
02487             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
02488             static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
02489             static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
02490             static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
02491             static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_)};
02492         return ValuesIn(array);
02493     }
02494
02495     ValueArray41(const ValueArray41& other) : v1_(other.v1_), v2_(other.v2_),
02496         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
02497         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
02498         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),

```

```

02499     v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
02500     v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
02501     v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
02502     v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
02503     v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
02504     v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
02505     v39_(other.v39_), v40_(other.v40_), v41_(other.v41_) {}
02506
02507 private:
02508 // No implementation - assignment is unsupported.
02509 void operator=(const ValueArray41& other);
02510
02511 const T1 v1_;
02512 const T2 v2_;
02513 const T3 v3_;
02514 const T4 v4_;
02515 const T5 v5_;
02516 const T6 v6_;
02517 const T7 v7_;
02518 const T8 v8_;
02519 const T9 v9_;
02520 const T10 v10_;
02521 const T11 v11_;
02522 const T12 v12_;
02523 const T13 v13_;
02524 const T14 v14_;
02525 const T15 v15_;
02526 const T16 v16_;
02527 const T17 v17_;
02528 const T18 v18_;
02529 const T19 v19_;
02530 const T20 v20_;
02531 const T21 v21_;
02532 const T22 v22_;
02533 const T23 v23_;
02534 const T24 v24_;
02535 const T25 v25_;
02536 const T26 v26_;
02537 const T27 v27_;
02538 const T28 v28_;
02539 const T29 v29_;
02540 const T30 v30_;
02541 const T31 v31_;
02542 const T32 v32_;
02543 const T33 v33_;
02544 const T34 v34_;
02545 const T35 v35_;
02546 const T36 v36_;
02547 const T37 v37_;
02548 const T38 v38_;
02549 const T39 v39_;
02550 const T40 v40_;
02551 const T41 v41_;
02552 };
02553
02554 template <typename T1, typename T2, typename T3, typename T4, typename T5,
02555   typename T6, typename T7, typename T8, typename T9, typename T10,
02556   typename T11, typename T12, typename T13, typename T14, typename T15,
02557   typename T16, typename T17, typename T18, typename T19, typename T20,
02558   typename T21, typename T22, typename T23, typename T24, typename T25,
02559   typename T26, typename T27, typename T28, typename T29, typename T30,
02560   typename T31, typename T32, typename T33, typename T34, typename T35,
02561   typename T36, typename T37, typename T38, typename T39, typename T40,
02562   typename T41, typename T42>
02563 class ValueArray42 {
02564 public:
02565     ValueArray42(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
02566                 T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
02567                 T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
02568                 T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
02569                 T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41,
02570                 T42 v42) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
02571                 v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
02572                 v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
02573                 v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26),
02574                 v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32),
02575                 v33_(v33), v34_(v34), v35_(v35), v36_(v36), v37_(v37), v38_(v38),
02576                 v39_(v39), v40_(v40), v41_(v41), v42_(v42) {}
02577
02578 template <typename T>
02579 operator ParamGenerator<T>() const {
02580     const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
02581                     static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
02582                     static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
02583                     static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
02584                     static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
02585                     static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_)};

```

```

02586     static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
02587     static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
02588     static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
02589     static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
02590     static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
02591     static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
02592     static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
02593     static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
02594     static_cast<T>(v42_)};
02595   return ValuesIn(array);
02596 }
02597
02598 ValueArray42(const ValueArray42& other) : v1_(other.v1_), v2_(other.v2_),
02599   v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
02600   v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
02601   v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
02602   v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
02603   v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
02604   v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
02605   v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
02606   v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
02607   v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
02608   v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_) {}
02609
02610 private:
02611 // No implementation - assignment is unsupported.
02612 void operator=(const ValueArray42& other);
02613
02614 const T1 v1_;
02615 const T2 v2_;
02616 const T3 v3_;
02617 const T4 v4_;
02618 const T5 v5_;
02619 const T6 v6_;
02620 const T7 v7_;
02621 const T8 v8_;
02622 const T9 v9_;
02623 const T10 v10_;
02624 const T11 v11_;
02625 const T12 v12_;
02626 const T13 v13_;
02627 const T14 v14_;
02628 const T15 v15_;
02629 const T16 v16_;
02630 const T17 v17_;
02631 const T18 v18_;
02632 const T19 v19_;
02633 const T20 v20_;
02634 const T21 v21_;
02635 const T22 v22_;
02636 const T23 v23_;
02637 const T24 v24_;
02638 const T25 v25_;
02639 const T26 v26_;
02640 const T27 v27_;
02641 const T28 v28_;
02642 const T29 v29_;
02643 const T30 v30_;
02644 const T31 v31_;
02645 const T32 v32_;
02646 const T33 v33_;
02647 const T34 v34_;
02648 const T35 v35_;
02649 const T36 v36_;
02650 const T37 v37_;
02651 const T38 v38_;
02652 const T39 v39_;
02653 const T40 v40_;
02654 const T41 v41_;
02655 const T42 v42_;
02656 };
02657
02658 template <typename T1, typename T2, typename T3, typename T4, typename T5,
02659   typename T6, typename T7, typename T8, typename T9, typename T10,
02660   typename T11, typename T12, typename T13, typename T14, typename T15,
02661   typename T16, typename T17, typename T18, typename T19, typename T20,
02662   typename T21, typename T22, typename T23, typename T24, typename T25,
02663   typename T26, typename T27, typename T28, typename T29, typename T30,
02664   typename T31, typename T32, typename T33, typename T34, typename T35,
02665   typename T36, typename T37, typename T38, typename T39, typename T40,
02666   typename T41, typename T42, typename T43>
02667 class ValueArray43 {
02668 public:
02669   ValueArray43(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
02670     T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
02671     T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
02672     T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,

```

```

02673     T34_v34, T35_v35, T36_v36, T37_v37, T38_v38, T39_v39, T40_v40, T41_v41,
02674     T42_v42, T43_v43) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6),
02675     v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13),
02676     v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19),
02677     v20_(v20), v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25),
02678     v26_(v26), v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31),
02679     v32_(v32), v33_(v33), v34_(v34), v35_(v35), v36_(v36), v37_(v37),
02680     v38_(v38), v39_(v39), v40_(v40), v41_(v41), v42_(v42), v43_(v43) {}  

02681  

02682     template <typename T>
02683     operator ParamGenerator<T>() const {
02684         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
02685             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
02686             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
02687             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
02688             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
02689             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
02690             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
02691             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
02692             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
02693             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
02694             static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
02695             static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
02696             static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
02697             static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
02698             static_cast<T>(v42_), static_cast<T>(v43_)};
02699         return ValuesIn(array);
02700     }
02701  

02702     ValueArray43(const ValueArray43& other) : v1_(other.v1_), v2_(other.v2_),
02703         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
02704         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
02705         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
02706         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
02707         v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
02708         v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
02709         v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
02710         v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
02711         v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
02712         v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
02713         v43_(other.v43_) {}  

02714  

02715     private:
02716     // No implementation - assignment is unsupported.
02717     void operator=(const ValueArray43& other);
02718  

02719     const T1 v1_;
02720     const T2 v2_;
02721     const T3 v3_;
02722     const T4 v4_;
02723     const T5 v5_;
02724     const T6 v6_;
02725     const T7 v7_;
02726     const T8 v8_;
02727     const T9 v9_;
02728     const T10 v10_;
02729     const T11 v11_;
02730     const T12 v12_;
02731     const T13 v13_;
02732     const T14 v14_;
02733     const T15 v15_;
02734     const T16 v16_;
02735     const T17 v17_;
02736     const T18 v18_;
02737     const T19 v19_;
02738     const T20 v20_;
02739     const T21 v21_;
02740     const T22 v22_;
02741     const T23 v23_;
02742     const T24 v24_;
02743     const T25 v25_;
02744     const T26 v26_;
02745     const T27 v27_;
02746     const T28 v28_;
02747     const T29 v29_;
02748     const T30 v30_;
02749     const T31 v31_;
02750     const T32 v32_;
02751     const T33 v33_;
02752     const T34 v34_;
02753     const T35 v35_;
02754     const T36 v36_;
02755     const T37 v37_;
02756     const T38 v38_;
02757     const T39 v39_;
02758     const T40 v40_;
02759     const T41 v41_;

```

```

02760     const T42 v42_;
02761     const T43 v43_;
02762 };
02763
02764 template <typename T1, typename T2, typename T3, typename T4, typename T5,
02765     typename T6, typename T7, typename T8, typename T9, typename T10,
02766     typename T11, typename T12, typename T13, typename T14, typename T15,
02767     typename T16, typename T17, typename T18, typename T19, typename T20,
02768     typename T21, typename T22, typename T23, typename T24, typename T25,
02769     typename T26, typename T27, typename T28, typename T29, typename T30,
02770     typename T31, typename T32, typename T33, typename T34, typename T35,
02771     typename T36, typename T37, typename T38, typename T39, typename T40,
02772     typename T41, typename T42, typename T43, typename T44>
02773 class ValueArray44 {
02774 public:
02775     ValueArray44(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
02776         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
02777         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
02778         T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
02779         T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41,
02780         T42 v42, T43 v43, T44 v44) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5),
02781         v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12),
02782         v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18),
02783         v19_(v19), v20_(v20), v21_(v21), v22_(v22), v23_(v23), v24_(v24),
02784         v25_(v25), v26_(v26), v27_(v27), v28_(v28), v29_(v29), v30_(v30),
02785         v31_(v31), v32_(v32), v33_(v33), v34_(v34), v35_(v35), v36_(v36),
02786         v37_(v37), v38_(v38), v39_(v39), v40_(v40), v41_(v41), v42_(v42),
02787         v43_(v43), v44_(v44) {}
02788
02789     template <typename T>
02790     operator ParamGenerator<T>() const {
02791         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
02792             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
02793             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
02794             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
02795             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
02796             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
02797             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
02798             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
02799             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
02800             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
02801             static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
02802             static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
02803             static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
02804             static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
02805             static_cast<T>(v42_), static_cast<T>(v43_), static_cast<T>(v44_)};
02806         return ValuesIn(array);
02807     }
02808
02809     ValueArray44(const ValueArray44& other) : v1_(other.v1_), v2_(other.v2_),
02810         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
02811         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
02812         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
02813         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
02814         v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
02815         v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
02816         v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
02817         v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
02818         v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
02819         v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
02820         v43_(other.v43_), v44_(other.v44_) {}
02821
02822 private:
02823 // No implementation - assignment is unsupported.
02824 void operator=(const ValueArray44& other);
02825
02826     const T1 v1_;
02827     const T2 v2_;
02828     const T3 v3_;
02829     const T4 v4_;
02830     const T5 v5_;
02831     const T6 v6_;
02832     const T7 v7_;
02833     const T8 v8_;
02834     const T9 v9_;
02835     const T10 v10_;
02836     const T11 v11_;
02837     const T12 v12_;
02838     const T13 v13_;
02839     const T14 v14_;
02840     const T15 v15_;
02841     const T16 v16_;
02842     const T17 v17_;
02843     const T18 v18_;
02844     const T19 v19_;
02845     const T20 v20_;
02846     const T21 v21_;

```

```

02847 const T22 v22_;
02848 const T23 v23_;
02849 const T24 v24_;
02850 const T25 v25_;
02851 const T26 v26_;
02852 const T27 v27_;
02853 const T28 v28_;
02854 const T29 v29_;
02855 const T30 v30_;
02856 const T31 v31_;
02857 const T32 v32_;
02858 const T33 v33_;
02859 const T34 v34_;
02860 const T35 v35_;
02861 const T36 v36_;
02862 const T37 v37_;
02863 const T38 v38_;
02864 const T39 v39_;
02865 const T40 v40_;
02866 const T41 v41_;
02867 const T42 v42_;
02868 const T43 v43_;
02869 const T44 v44_;
02870 };
02871
02872 template <typename T1, typename T2, typename T3, typename T4, typename T5,
02873     typename T6, typename T7, typename T8, typename T9, typename T10,
02874     typename T11, typename T12, typename T13, typename T14, typename T15,
02875     typename T16, typename T17, typename T18, typename T19, typename T20,
02876     typename T21, typename T22, typename T23, typename T24, typename T25,
02877     typename T26, typename T27, typename T28, typename T29, typename T30,
02878     typename T31, typename T32, typename T33, typename T34, typename T35,
02879     typename T36, typename T37, typename T38, typename T39, typename T40,
02880     typename T41, typename T42, typename T43, typename T44, typename T45>
02881 class ValueArray45 {
02882 public:
02883     ValueArray45(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
02884         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
02885         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
02886         T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
02887         T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41,
02888         T42 v42, T43 v43, T44 v44, T45 v45) : v1_(v1), v2_(v2), v3_(v3), v4_(v4),
02889         v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11),
02890         v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17),
02891         v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22), v23_(v23),
02892         v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28), v29_(v29),
02893         v30_(v30), v31_(v31), v32_(v32), v33_(v33), v34_(v34), v35_(v35),
02894         v36_(v36), v37_(v37), v38_(v38), v39_(v39), v40_(v40), v41_(v41),
02895         v42_(v42), v43_(v43), v44_(v44), v45_(v45) {}
02896
02897 template <typename T>
02898 operator ParamGenerator<T>() const {
02899     const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
02900         static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
02901         static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
02902         static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
02903         static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
02904         static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
02905         static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
02906         static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
02907         static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
02908         static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
02909         static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
02910         static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
02911         static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
02912         static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
02913         static_cast<T>(v42_), static_cast<T>(v43_), static_cast<T>(v44_),
02914         static_cast<T>(v45_)};
02915     return ValuesIn(array);
02916 }
02917
02918 ValueArray45(const ValueArray45& other) : v1_(other.v1_), v2_(other.v2_),
02919     v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
02920     v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
02921     v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
02922     v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
02923     v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
02924     v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
02925     v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
02926     v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
02927     v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
02928     v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
02929     v43_(other.v43_), v44_(other.v44_), v45_(other.v45_) {}
02930
02931 private:
02932 // No implementation - assignment is unsupported.
02933 void operator=(const ValueArray45& other);

```

```

02934
02935     const T1 v1_;
02936     const T2 v2_;
02937     const T3 v3_;
02938     const T4 v4_;
02939     const T5 v5_;
02940     const T6 v6_;
02941     const T7 v7_;
02942     const T8 v8_;
02943     const T9 v9_;
02944     const T10 v10_;
02945     const T11 v11_;
02946     const T12 v12_;
02947     const T13 v13_;
02948     const T14 v14_;
02949     const T15 v15_;
02950     const T16 v16_;
02951     const T17 v17_;
02952     const T18 v18_;
02953     const T19 v19_;
02954     const T20 v20_;
02955     const T21 v21_;
02956     const T22 v22_;
02957     const T23 v23_;
02958     const T24 v24_;
02959     const T25 v25_;
02960     const T26 v26_;
02961     const T27 v27_;
02962     const T28 v28_;
02963     const T29 v29_;
02964     const T30 v30_;
02965     const T31 v31_;
02966     const T32 v32_;
02967     const T33 v33_;
02968     const T34 v34_;
02969     const T35 v35_;
02970     const T36 v36_;
02971     const T37 v37_;
02972     const T38 v38_;
02973     const T39 v39_;
02974     const T40 v40_;
02975     const T41 v41_;
02976     const T42 v42_;
02977     const T43 v43_;
02978     const T44 v44_;
02979     const T45 v45_;
02980 };
02981
02982 template <typename T1, typename T2, typename T3, typename T4, typename T5,
02983     typename T6, typename T7, typename T8, typename T9, typename T10,
02984     typename T11, typename T12, typename T13, typename T14, typename T15,
02985     typename T16, typename T17, typename T18, typename T19, typename T20,
02986     typename T21, typename T22, typename T23, typename T24, typename T25,
02987     typename T26, typename T27, typename T28, typename T29, typename T30,
02988     typename T31, typename T32, typename T33, typename T34, typename T35,
02989     typename T36, typename T37, typename T38, typename T39, typename T40,
02990     typename T41, typename T42, typename T43, typename T44, typename T45,
02991     typename T46>
02992 class ValueArray46 {
02993 public:
02994     ValueArray46(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
02995         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
02996         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
02997         T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
02998         T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41,
02999         T42 v42, T43 v43, T44 v44, T45 v45, T46 v46) : v1_(v1), v2_(v2), v3_(v3),
03000         v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
03001         v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
03002         v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22),
03003         v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28),
03004         v29_(v29), v30_(v30), v31_(v31), v32_(v32), v33_(v33), v34_(v34),
03005         v35_(v35), v36_(v36), v37_(v37), v38_(v38), v39_(v39), v40_(v40),
03006         v41_(v41), v42_(v42), v43_(v43), v44_(v44), v45_(v45), v46_(v46) {}
03007
03008 template <typename T>
03009 operator ParamGenerator<T>() const {
03010     const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
03011         static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
03012         static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
03013         static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
03014         static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
03015         static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
03016         static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
03017         static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
03018         static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
03019         static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
03020         static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_)};

```

```

03021     static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
03022     static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
03023     static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
03024     static_cast<T>(v42_), static_cast<T>(v43_), static_cast<T>(v44_),
03025     static_cast<T>(v45_), static_cast<T>(v46_)};
03026     return ValuesIn(array);
03027 }
03028
03029 ValueArray46(const ValueArray46& other) : v1_(other.v1_), v2_(other.v2_),
03030     v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
03031     v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
03032     v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
03033     v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
03034     v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
03035     v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
03036     v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
03037     v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
03038     v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
03039     v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
03040     v43_(other.v43_), v44_(other.v44_), v45_(other.v45_), v46_(other.v46_) {}
03041
03042 private:
03043 // No implementation - assignment is unsupported.
03044 void operator=(const ValueArray46& other);
03045
03046 const T1 v1_;
03047 const T2 v2_;
03048 const T3 v3_;
03049 const T4 v4_;
03050 const T5 v5_;
03051 const T6 v6_;
03052 const T7 v7_;
03053 const T8 v8_;
03054 const T9 v9_;
03055 const T10 v10_;
03056 const T11 v11_;
03057 const T12 v12_;
03058 const T13 v13_;
03059 const T14 v14_;
03060 const T15 v15_;
03061 const T16 v16_;
03062 const T17 v17_;
03063 const T18 v18_;
03064 const T19 v19_;
03065 const T20 v20_;
03066 const T21 v21_;
03067 const T22 v22_;
03068 const T23 v23_;
03069 const T24 v24_;
03070 const T25 v25_;
03071 const T26 v26_;
03072 const T27 v27_;
03073 const T28 v28_;
03074 const T29 v29_;
03075 const T30 v30_;
03076 const T31 v31_;
03077 const T32 v32_;
03078 const T33 v33_;
03079 const T34 v34_;
03080 const T35 v35_;
03081 const T36 v36_;
03082 const T37 v37_;
03083 const T38 v38_;
03084 const T39 v39_;
03085 const T40 v40_;
03086 const T41 v41_;
03087 const T42 v42_;
03088 const T43 v43_;
03089 const T44 v44_;
03090 const T45 v45_;
03091 const T46 v46_;
03092 };
03093
03094 template <typename T1, typename T2, typename T3, typename T4, typename T5,
03095     typename T6, typename T7, typename T8, typename T9, typename T10,
03096     typename T11, typename T12, typename T13, typename T14, typename T15,
03097     typename T16, typename T17, typename T18, typename T19, typename T20,
03098     typename T21, typename T22, typename T23, typename T24, typename T25,
03099     typename T26, typename T27, typename T28, typename T29, typename T30,
03100     typename T31, typename T32, typename T33, typename T34, typename T35,
03101     typename T36, typename T37, typename T38, typename T39, typename T40,
03102     typename T41, typename T42, typename T43, typename T44, typename T45,
03103     typename T46, typename T47>
03104 class ValueArray47 {
03105 public:
03106     ValueArray47(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
03107                 T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
```

```

03108     T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
03109     T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
03110     T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41,
03111     T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47) : v1_(v1), v2_(v2),
03112     v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
03113     v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
03114     v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22),
03115     v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28),
03116     v29_(v29), v30_(v30), v31_(v31), v32_(v32), v33_(v33), v34_(v34),
03117     v35_(v35), v36_(v36), v37_(v37), v38_(v38), v39_(v39), v40_(v40),
03118     v41_(v41), v42_(v42), v43_(v43), v44_(v44), v45_(v45), v46_(v46),
03119     v47_(v47) {}
03120
03121     template <typename T>
03122     operator ParamGenerator<T>() const {
03123         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
03124             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
03125             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
03126             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
03127             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
03128             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
03129             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
03130             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
03131             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
03132             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
03133             static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
03134             static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
03135             static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
03136             static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
03137             static_cast<T>(v42_), static_cast<T>(v43_), static_cast<T>(v44_),
03138             static_cast<T>(v45_), static_cast<T>(v46_), static_cast<T>(v47_)};
03139         return ValuesIn(array);
03140     }
03141
03142     ValueArray47(const ValueArray47& other) : v1_(other.v1_), v2_(other.v2_),
03143         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
03144         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
03145         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
03146         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
03147         v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
03148         v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
03149         v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
03150         v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
03151         v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
03152         v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
03153         v43_(other.v43_), v44_(other.v44_), v45_(other.v45_), v46_(other.v46_),
03154         v47_(other.v47_) {}
03155
03156     private:
03157     // No implementation - assignment is unsupported.
03158     void operator=(const ValueArray47& other);
03159
03160     const T1 v1_;
03161     const T2 v2_;
03162     const T3 v3_;
03163     const T4 v4_;
03164     const T5 v5_;
03165     const T6 v6_;
03166     const T7 v7_;
03167     const T8 v8_;
03168     const T9 v9_;
03169     const T10 v10_;
03170     const T11 v11_;
03171     const T12 v12_;
03172     const T13 v13_;
03173     const T14 v14_;
03174     const T15 v15_;
03175     const T16 v16_;
03176     const T17 v17_;
03177     const T18 v18_;
03178     const T19 v19_;
03179     const T20 v20_;
03180     const T21 v21_;
03181     const T22 v22_;
03182     const T23 v23_;
03183     const T24 v24_;
03184     const T25 v25_;
03185     const T26 v26_;
03186     const T27 v27_;
03187     const T28 v28_;
03188     const T29 v29_;
03189     const T30 v30_;
03190     const T31 v31_;
03191     const T32 v32_;
03192     const T33 v33_;
03193     const T34 v34_;
03194     const T35 v35_;

```

```

03195 const T36 v36_;
03196 const T37 v37_;
03197 const T38 v38_;
03198 const T39 v39_;
03199 const T40 v40_;
03200 const T41 v41_;
03201 const T42 v42_;
03202 const T43 v43_;
03203 const T44 v44_;
03204 const T45 v45_;
03205 const T46 v46_;
03206 const T47 v47_;
03207 };
03208
03209 template <typename T1, typename T2, typename T3, typename T4, typename T5,
03210     typename T6, typename T7, typename T8, typename T9, typename T10,
03211     typename T11, typename T12, typename T13, typename T14, typename T15,
03212     typename T16, typename T17, typename T18, typename T19, typename T20,
03213     typename T21, typename T22, typename T23, typename T24, typename T25,
03214     typename T26, typename T27, typename T28, typename T29, typename T30,
03215     typename T31, typename T32, typename T33, typename T34, typename T35,
03216     typename T36, typename T37, typename T38, typename T39, typename T40,
03217     typename T41, typename T42, typename T43, typename T44, typename T45,
03218     typename T46, typename T47, typename T48>
03219 class ValueArray48 {
03220 public:
03221     ValueArray48(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
03222         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
03223         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
03224         T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
03225         T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41,
03226         T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47, T48 v48) : v1_(v1),
03227         v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9),
03228         v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15),
03229         v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21),
03230         v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27),
03231         v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32), v33_(v33),
03232         v34_(v34), v35_(v35), v36_(v36), v37_(v37), v38_(v38), v39_(v39),
03233         v40_(v40), v41_(v41), v42_(v42), v43_(v43), v44_(v44), v45_(v45),
03234         v46_(v46), v47_(v47), v48_(v48) {}
03235
03236     template <typename T>
03237     operator ParamGenerator<T>() const {
03238         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
03239             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
03240             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
03241             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
03242             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
03243             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
03244             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
03245             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
03246             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
03247             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
03248             static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
03249             static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
03250             static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
03251             static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
03252             static_cast<T>(v42_), static_cast<T>(v43_), static_cast<T>(v44_),
03253             static_cast<T>(v45_), static_cast<T>(v46_), static_cast<T>(v47_),
03254             static_cast<T>(v48_)};
03255         return ValuesIn(array);
03256     }
03257
03258     ValueArray48(const ValueArray48& other) : v1_(other.v1_), v2_(other.v2_),
03259         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
03260         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
03261         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
03262         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
03263         v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
03264         v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
03265         v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
03266         v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
03267         v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
03268         v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
03269         v43_(other.v43_), v44_(other.v44_), v45_(other.v45_), v46_(other.v46_),
03270         v47_(other.v47_), v48_(other.v48_) {}
03271
03272     private:
03273     // No implementation - assignment is unsupported.
03274     void operator=(const ValueArray48& other);
03275
03276     const T1 v1_;
03277     const T2 v2_;
03278     const T3 v3_;
03279     const T4 v4_;
03280     const T5 v5_;
03281     const T6 v6_;

```

```

03282 const T7 v7_;
03283 const T8 v8_;
03284 const T9 v9_;
03285 const T10 v10_;
03286 const T11 v11_;
03287 const T12 v12_;
03288 const T13 v13_;
03289 const T14 v14_;
03290 const T15 v15_;
03291 const T16 v16_;
03292 const T17 v17_;
03293 const T18 v18_;
03294 const T19 v19_;
03295 const T20 v20_;
03296 const T21 v21_;
03297 const T22 v22_;
03298 const T23 v23_;
03299 const T24 v24_;
03300 const T25 v25_;
03301 const T26 v26_;
03302 const T27 v27_;
03303 const T28 v28_;
03304 const T29 v29_;
03305 const T30 v30_;
03306 const T31 v31_;
03307 const T32 v32_;
03308 const T33 v33_;
03309 const T34 v34_;
03310 const T35 v35_;
03311 const T36 v36_;
03312 const T37 v37_;
03313 const T38 v38_;
03314 const T39 v39_;
03315 const T40 v40_;
03316 const T41 v41_;
03317 const T42 v42_;
03318 const T43 v43_;
03319 const T44 v44_;
03320 const T45 v45_;
03321 const T46 v46_;
03322 const T47 v47_;
03323 const T48 v48_;
03324 };
03325
03326 template <typename T1, typename T2, typename T3, typename T4, typename T5,
03327     typename T6, typename T7, typename T8, typename T9, typename T10,
03328     typename T11, typename T12, typename T13, typename T14, typename T15,
03329     typename T16, typename T17, typename T18, typename T19, typename T20,
03330     typename T21, typename T22, typename T23, typename T24, typename T25,
03331     typename T26, typename T27, typename T28, typename T29, typename T30,
03332     typename T31, typename T32, typename T33, typename T34, typename T35,
03333     typename T36, typename T37, typename T38, typename T39, typename T40,
03334     typename T41, typename T42, typename T43, typename T44, typename T45,
03335     typename T46, typename T47, typename T48, typename T49>
03336 class ValueArray49 {
03337 public:
03338     ValueArray49(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
03339         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
03340         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
03341         T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
03342         T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41,
03343         T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47, T48 v48,
03344         T49 v49) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
03345         v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
03346         v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
03347         v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26),
03348         v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32),
03349         v33_(v33), v34_(v34), v35_(v35), v36_(v36), v37_(v37), v38_(v38),
03350         v39_(v39), v40_(v40), v41_(v41), v42_(v42), v43_(v43), v44_(v44),
03351         v45_(v45), v46_(v46), v47_(v47), v48_(v48), v49_(v49) {}
03352
03353     template <typename T>
03354     operator ParamGenerator<T>() const {
03355         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
03356             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
03357             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
03358             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
03359             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
03360             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
03361             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
03362             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
03363             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
03364             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
03365             static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
03366             static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
03367             static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
03368             static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_)};

```

```

03369     static_cast<T>(v42_), static_cast<T>(v43_), static_cast<T>(v44_),
03370     static_cast<T>(v45_), static_cast<T>(v46_), static_cast<T>(v47_),
03371     static_cast<T>(v48_), static_cast<T>(v49_));
03372     return ValuesIn(array);
03373 }
03374
03375 ValueArray49(const ValueArray49& other) : v1_(other.v1_), v2_(other.v2_),
03376     v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
03377     v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
03378     v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
03379     v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
03380     v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
03381     v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
03382     v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
03383     v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
03384     v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
03385     v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
03386     v43_(other.v43_), v44_(other.v44_), v45_(other.v45_), v46_(other.v46_),
03387     v47_(other.v47_), v48_(other.v48_), v49_(other.v49_) {}
03388
03389 private:
03390 // No implementation - assignment is unsupported.
03391 void operator=(const ValueArray49& other);
03392
03393 const T1 v1_;
03394 const T2 v2_;
03395 const T3 v3_;
03396 const T4 v4_;
03397 const T5 v5_;
03398 const T6 v6_;
03399 const T7 v7_;
03400 const T8 v8_;
03401 const T9 v9_;
03402 const T10 v10_;
03403 const T11 v11_;
03404 const T12 v12_;
03405 const T13 v13_;
03406 const T14 v14_;
03407 const T15 v15_;
03408 const T16 v16_;
03409 const T17 v17_;
03410 const T18 v18_;
03411 const T19 v19_;
03412 const T20 v20_;
03413 const T21 v21_;
03414 const T22 v22_;
03415 const T23 v23_;
03416 const T24 v24_;
03417 const T25 v25_;
03418 const T26 v26_;
03419 const T27 v27_;
03420 const T28 v28_;
03421 const T29 v29_;
03422 const T30 v30_;
03423 const T31 v31_;
03424 const T32 v32_;
03425 const T33 v33_;
03426 const T34 v34_;
03427 const T35 v35_;
03428 const T36 v36_;
03429 const T37 v37_;
03430 const T38 v38_;
03431 const T39 v39_;
03432 const T40 v40_;
03433 const T41 v41_;
03434 const T42 v42_;
03435 const T43 v43_;
03436 const T44 v44_;
03437 const T45 v45_;
03438 const T46 v46_;
03439 const T47 v47_;
03440 const T48 v48_;
03441 const T49 v49_;
03442 };
03443
03444 template <typename T1, typename T2, typename T3, typename T4, typename T5,
03445     typename T6, typename T7, typename T8, typename T9, typename T10,
03446     typename T11, typename T12, typename T13, typename T14, typename T15,
03447     typename T16, typename T17, typename T18, typename T19, typename T20,
03448     typename T21, typename T22, typename T23, typename T24, typename T25,
03449     typename T26, typename T27, typename T28, typename T29, typename T30,
03450     typename T31, typename T32, typename T33, typename T34, typename T35,
03451     typename T36, typename T37, typename T38, typename T39, typename T40,
03452     typename T41, typename T42, typename T43, typename T44, typename T45,
03453     typename T46, typename T47, typename T48, typename T49, typename T50>
03454 class ValueArray50 {
03455 public:

```

```

03456     ValueArray50(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
03457         T10 v10, T11 v11, T12 v12, T13 v13, T14 v14, T15 v15, T16 v16, T17 v17,
03458         T18 v18, T19 v19, T20 v20, T21 v21, T22 v22, T23 v23, T24 v24, T25 v25,
03459         T26 v26, T27 v27, T28 v28, T29 v29, T30 v30, T31 v31, T32 v32, T33 v33,
03460         T34 v34, T35 v35, T36 v36, T37 v37, T38 v38, T39 v39, T40 v40, T41 v41,
03461         T42 v42, T43 v43, T44 v44, T45 v45, T46 v46, T47 v47, T48 v48, T49 v49,
03462         T50 v50) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
03463         v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
03464         v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
03465         v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26),
03466         v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32),
03467         v33_(v33), v34_(v34), v35_(v35), v36_(v36), v37_(v37), v38_(v38),
03468         v39_(v39), v40_(v40), v41_(v41), v42_(v42), v43_(v43), v44_(v44),
03469         v45_(v45), v46_(v46), v47_(v47), v48_(v48), v49_(v49), v50_(v50) {}
03470
03471     template <typename T>
03472     operator ParamGenerator<T>() const {
03473         const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
03474             static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
03475             static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
03476             static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
03477             static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
03478             static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
03479             static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
03480             static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
03481             static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
03482             static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
03483             static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
03484             static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
03485             static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
03486             static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
03487             static_cast<T>(v42_), static_cast<T>(v43_), static_cast<T>(v44_),
03488             static_cast<T>(v45_), static_cast<T>(v46_), static_cast<T>(v47_),
03489             static_cast<T>(v48_), static_cast<T>(v49_), static_cast<T>(v50_)};
03490         return ValuesIn(array);
03491     }
03492
03493     ValueArray50(const ValueArray50& other) : v1_(other.v1_), v2_(other.v2_),
03494         v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
03495         v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
03496         v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
03497         v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
03498         v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
03499         v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
03500         v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
03501         v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
03502         v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
03503         v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
03504         v43_(other.v43_), v44_(other.v44_), v45_(other.v45_), v46_(other.v46_),
03505         v47_(other.v47_), v48_(other.v48_), v49_(other.v49_), v50_(other.v50_) {}
03506
03507     private:
03508     // No implementation - assignment is unsupported.
03509     void operator=(const ValueArray50& other);
03510
03511     const T1 v1_;
03512     const T2 v2_;
03513     const T3 v3_;
03514     const T4 v4_;
03515     const T5 v5_;
03516     const T6 v6_;
03517     const T7 v7_;
03518     const T8 v8_;
03519     const T9 v9_;
03520     const T10 v10_;
03521     const T11 v11_;
03522     const T12 v12_;
03523     const T13 v13_;
03524     const T14 v14_;
03525     const T15 v15_;
03526     const T16 v16_;
03527     const T17 v17_;
03528     const T18 v18_;
03529     const T19 v19_;
03530     const T20 v20_;
03531     const T21 v21_;
03532     const T22 v22_;
03533     const T23 v23_;
03534     const T24 v24_;
03535     const T25 v25_;
03536     const T26 v26_;
03537     const T27 v27_;
03538     const T28 v28_;
03539     const T29 v29_;
03540     const T30 v30_;
03541     const T31 v31_;
03542     const T32 v32_;

```

```

03543     const T33 v33_;
03544     const T34 v34_;
03545     const T35 v35_;
03546     const T36 v36_;
03547     const T37 v37_;
03548     const T38 v38_;
03549     const T39 v39_;
03550     const T40 v40_;
03551     const T41 v41_;
03552     const T42 v42_;
03553     const T43 v43_;
03554     const T44 v44_;
03555     const T45 v45_;
03556     const T46 v46_;
03557     const T47 v47_;
03558     const T48 v48_;
03559     const T49 v49_;
03560     const T50 v50_;
03561 };
03562
03563 # if GTEST_HAS_COMBINE
03564 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
03565 //
03566 // Generates values from the Cartesian product of values produced
03567 // by the argument generators.
03568 //
03569 template <typename T1, typename T2>
03570 class CartesianProductGenerator2
03571     : public ParamGeneratorInterface<::testing::tuple<T1, T2> > {
03572 public:
03573     typedef ::testing::tuple<T1, T2> ParamType;
03574
03575     CartesianProductGenerator2(const ParamGenerator<T1>& g1,
03576         const ParamGenerator<T2>& g2)
03577         : g1_(g1), g2_(g2) {}
03578     virtual ~CartesianProductGenerator2() {}

03579     virtual ParamIteratorInterface<ParamType>* Begin() const {
03580         return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin());
03581     }
03582     virtual ParamIteratorInterface<ParamType>* End() const {
03583         return new Iterator(this, g1_, g1_.end(), g2_, g2_.end());
03584     }
03585 }
03586
03587 private:
03588     class Iterator : public ParamIteratorInterface<ParamType> {
03589     public:
03590         Iterator(const ParamGeneratorInterface<ParamType>* base,
03591             const ParamGenerator<T1>& g1,
03592             const typename ParamGenerator<T1>::iterator& current1,
03593             const ParamGenerator<T2>& g2,
03594             const typename ParamGenerator<T2>::iterator& current2)
03595             : base_(base),
03596                 begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
03597                 begin2_(g2.begin()), end2_(g2.end()), current2_(current2) {
03598             ComputeCurrentValue();
03599         }
03600         virtual ~Iterator() {}

03601         virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
03602             return base_;
03603         }
03604         // Advance should not be called on beyond-of-range iterators
03605         // so no component iterators must be beyond end of range, either.
03606         virtual void Advance() {
03607             assert(!AtEnd());
03608             ++current2_;
03609             if (current2_ == end2_) {
03610                 current2_ = begin2_;
03611                 ++current1_;
03612             }
03613             ComputeCurrentValue();
03614         }
03615         virtual ParamIteratorInterface<ParamType>* Clone() const {
03616             return new Iterator(*this);
03617         }
03618         virtual const ParamType* Current() const { return current_value_.get(); }
03619         virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
03620             // Having the same base generator guarantees that the other
03621             // iterator is of the same type and we can downcast.
03622             GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
03623             « "The program attempted to compare iterators "
03624             « "from different generators." « std::endl;
03625             const Iterator* typed_other =
03626                 CheckedDowncastToActualType<const Iterator>(&other);
03627             // We must report iterators equal if they both point beyond their
03628             // respective ranges. That can happen in a variety of fashions,
03629

```

```

03630     // so we have to consult AtEnd().
03631     return (AtEnd() && typed_other->AtEnd()) ||
03632         (
03633             current1_ == typed_other->current1_ &&
03634             current2_ == typed_other->current2_);
03635     }
03636
03637 private:
03638     Iterator(const Iterator& other)
03639         : base_(other.base_),
03640         begin1_(other.begin1_),
03641         endl_(other.end1_),
03642         current1_(other.current1_),
03643         begin2_(other.begin2_),
03644         end2_(other.end2_),
03645         current2_(other.current2_) {
03646     ComputeCurrentValue();
03647 }
03648
03649     void ComputeCurrentValue() {
03650         if (!AtEnd())
03651             current_value_.reset(new ParamType(*current1_, *current2_));
03652     }
03653     bool AtEnd() const {
03654         // We must report iterator past the end of the range when either of the
03655         // component iterators has reached the end of its range.
03656         return
03657             current1_ == endl_ ||
03658             current2_ == end2_;
03659     }
03660
03661     // No implementation - assignment is unsupported.
03662     void operator=(const Iterator& other);
03663
03664     const ParamGeneratorInterface<ParamType>* const base_;
03665     // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
03666     // current[i]_ is the actual traversing iterator.
03667     const typename ParamGenerator<T1>::iterator begin1_;
03668     const typename ParamGenerator<T1>::iterator endl_;
03669     typename ParamGenerator<T1>::iterator current1_;
03670     const typename ParamGenerator<T2>::iterator begin2_;
03671     const typename ParamGenerator<T2>::iterator end2_;
03672     typename ParamGenerator<T2>::iterator current2_;
03673     linked_ptr<ParamType> current_value_;
03674 }; // class CartesianProductGenerator2::Iterator
03675
03676     // No implementation - assignment is unsupported.
03677     void operator=(const CartesianProductGenerator2& other);
03678
03679     const ParamGenerator<T1> g1_;
03680     const ParamGenerator<T2> g2_;
03681 }; // class CartesianProductGenerator2
03682
03683
03684 template <typename T1, typename T2, typename T3>
03685 class CartesianProductGenerator3
03686     : public ParamGeneratorInterface<::testing::tuple<T1, T2, T3> > {
03687     public:
03688         typedef ::testing::tuple<T1, T2, T3> ParamType;
03689
03690         CartesianProductGenerator3(const ParamGenerator<T1>& g1,
03691             const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3)
03692             : g1_(g1), g2_(g2), g3_(g3) {}
03693         virtual ~CartesianProductGenerator3() {}
03694
03695         virtual ParamIteratorInterface<ParamType>* Begin() const {
03696             return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin(), g3_,
03697                 g3_.begin());
03698         }
03699         virtual ParamIteratorInterface<ParamType>* End() const {
03700             return new Iterator(this, g1_, g1_.end(), g2_, g2_.end(), g3_, g3_.end());
03701         }
03702
03703     private:
03704         class Iterator : public ParamIteratorInterface<ParamType> {
03705         public:
03706             Iterator(const ParamGeneratorInterface<ParamType>* base,
03707                 const ParamGenerator<T1>& g1,
03708                 const typename ParamGenerator<T1>::iterator& current1,
03709                 const ParamGenerator<T2>& g2,
03710                 const typename ParamGenerator<T2>::iterator& current2,
03711                 const ParamGenerator<T3>& g3,
03712                 const typename ParamGenerator<T3>::iterator& current3)
03713                 : base_(base),
03714                     begin1_(g1.begin()), endl_(g1.end()), current1_(current1),
03715                     begin2_(g2.begin()), end2_(g2.end()), current2_(current2),
03716                     begin3_(g3.begin()), end3_(g3.end()), current3_(current3) {

```

```

03717     ComputeCurrentValue();
03718 }
03719 virtual ~Iterator() {}
03720
03721 virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
03722     return base_;
03723 }
03724 // Advance should not be called on beyond-of-range iterators
03725 // so no component iterators must be beyond end of range, either.
03726 virtual void Advance() {
03727     assert(!AtEnd());
03728     ++current3_;
03729     if (current3_ == end3_) {
03730         current3_ = begin3_;
03731         ++current2_;
03732     }
03733     if (current2_ == end2_) {
03734         current2_ = begin2_;
03735         ++current1_;
03736     }
03737     ComputeCurrentValue();
03738 }
03739 virtual ParamIteratorInterface<ParamType>* Clone() const {
03740     return new Iterator(*this);
03741 }
03742 virtual const ParamType* Current() const { return current_value_.get(); }
03743 virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
03744     // Having the same base generator guarantees that the other
03745     // iterator is of the same type and we can downcast.
03746     GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
03747         << "The program attempted to compare iterators "
03748         << "from different generators." << std::endl;
03749     const Iator* typed_other =
03750         CheckedDowncastToActualType<const Iator>(&other);
03751     // We must report iterators equal if they both point beyond their
03752     // respective ranges. That can happen in a variety of fashions,
03753     // so we have to consult AtEnd().
03754     return (AtEnd() && typed_other->AtEnd()) ||
03755     (
03756         current1_ == typed_other->current1_ &&
03757         current2_ == typed_other->current2_ &&
03758         current3_ == typed_other->current3_);
03759 }
03760
03761 private:
03762     Iator(const Iator& other)
03763         : base_(other.base_),
03764         begin1_(other.begin1_),
03765         endl_(other.endl_),
03766         current1_(other.current1_),
03767         begin2_(other.begin2_),
03768         end2_(other.end2_),
03769         current2_(other.current2_),
03770         begin3_(other.begin3_),
03771         end3_(other.end3_),
03772         current3_(other.current3_) {
03773     ComputeCurrentValue();
03774 }
03775
03776 void ComputeCurrentValue() {
03777     if (!AtEnd())
03778         current_value_.reset(new ParamType(*current1_, *current2_, *current3_));
03779 }
03780 bool AtEnd() const {
03781     // We must report iterator past the end of the range when either of the
03782     // component iterators has reached the end of its range.
03783     return
03784         current1_ == endl_ ||
03785         current2_ == end2_ ||
03786         current3_ == end3_;
03787 }
03788
03789 // No implementation - assignment is unsupported.
03790 void operator=(const Iator& other);
03791
03792 const ParamGeneratorInterface<ParamType>* const base_;
03793 // begin[i]_ and end[i]_ define the i-th range that Iator traverses.
03794 // current[i]_ is the actual traversing iterator.
03795 const typename ParamGenerator<T1>::iterator begin1_;
03796 const typename ParamGenerator<T1>::iterator endl_;
03797 typename ParamGenerator<T1>::iterator current1_;
03798 const typename ParamGenerator<T2>::iterator begin2_;
03799 const typename ParamGenerator<T2>::iterator endl2_;
03800 typename ParamGenerator<T2>::iterator current2_;
03801 const typename ParamGenerator<T3>::iterator begin3_;
03802 const typename ParamGenerator<T3>::iterator endl3_;
03803 typename ParamGenerator<T3>::iterator current3_;

```

```

03804     linked_ptr<ParamType> current_value_;
03805 }; // class CartesianProductGenerator3::Iterator
03806
03807 // No implementation - assignment is unsupported.
03808 void operator=(const CartesianProductGenerator3& other);
03809
03810 const ParamGenerator<T1> g1_;
03811 const ParamGenerator<T2> g2_;
03812 const ParamGenerator<T3> g3_;
03813 }; // class CartesianProductGenerator3
03814
03815
03816 template <typename T1, typename T2, typename T3, typename T4>
03817 class CartesianProductGenerator4
03818   : public ParamGeneratorInterface<::testing::tuple<T1, T2, T3, T4> > {
03819 public:
03820   typedef ::testing::tuple<T1, T2, T3, T4> ParamType;
03821
03822   CartesianProductGenerator4(const ParamGenerator<T1>& g1,
03823     const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3,
03824     const ParamGenerator<T4>& g4)
03825     : g1_(g1), g2_(g2), g3_(g3), g4_(g4) {}
03826   virtual ~CartesianProductGenerator4() {}
03827
03828   virtual ParamIteratorInterface<ParamType>* Begin() const {
03829     return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin(), g3_,
03830       g3_.begin(), g4_, g4_.begin());
03831   }
03832   virtual ParamIteratorInterface<ParamType>* End() const {
03833     return new Iterator(this, g1_, g1_.end(), g2_, g2_.end(), g3_,
03834       g3_.end(), g4_, g4_.end());
03835   }
03836
03837 private:
03838   class Iterator : public ParamIteratorInterface<ParamType> {
03839   public:
03840     Iterator(const ParamGeneratorInterface<ParamType>* base,
03841       const ParamGenerator<T1>& g1,
03842       const typename ParamGenerator<T1>::iterator& current1,
03843       const ParamGenerator<T2>& g2,
03844       const typename ParamGenerator<T2>::iterator& current2,
03845       const ParamGenerator<T3>& g3,
03846       const typename ParamGenerator<T3>::iterator& current3,
03847       const ParamGenerator<T4>& g4,
03848       const typename ParamGenerator<T4>::iterator& current4)
03849       : base_(base),
03850         begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
03851         begin2_(g2.begin()), end2_(g2.end()), current2_(current2),
03852         begin3_(g3.begin()), end3_(g3.end()), current3_(current3),
03853         begin4_(g4.begin()), end4_(g4.end()), current4_(current4) {
03854       ComputeCurrentValue();
03855     }
03856   virtual ~Iterator() {}
03857
03858   virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
03859     return base_;
03860   }
03861   // Advance should not be called on beyond-of-range iterators
03862   // so no component iterators must be beyond end of range, either.
03863   virtual void Advance() {
03864     assert(!AtEnd());
03865     ++current4_;
03866     if (current4_ == end4_) {
03867       current4_ = begin4_;
03868       ++current3_;
03869     }
03870     if (current3_ == end3_) {
03871       current3_ = begin3_;
03872       ++current2_;
03873     }
03874     if (current2_ == end2_) {
03875       current2_ = begin2_;
03876       ++current1_;
03877     }
03878     ComputeCurrentValue();
03879   }
03880   virtual ParamIteratorInterface<ParamType>* Clone() const {
03881     return new Iterator(*this);
03882   }
03883   virtual const ParamType* Current() const { return current_value_.get(); }
03884   virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
03885     // Having the same base generator guarantees that the other
03886     // iterator is of the same type and we can downcast.
03887     GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
03888     « "The program attempted to compare iterators "
03889     « "from different generators." « std::endl;
03890     const Iterator* typed_other =

```

```

03891     CheckedDowncastToActualType<const Iterator>(&other);
03892     // We must report iterators equal if they both point beyond their
03893     // respective ranges. That can happen in a variety of fashions,
03894     // so we have to consult AtEnd().
03895     return (AtEnd() && typed_other->AtEnd()) ||
03896     (
03897         current1_ == typed_other->current1_ &&
03898         current2_ == typed_other->current2_ &&
03899         current3_ == typed_other->current3_ &&
03900         current4_ == typed_other->current4_);
03901     }
03902
03903 private:
03904     Iterator(const Iterator& other)
03905     : base_(other.base_),
03906     begin1_(other.begin1_),
03907     endl_(other.endl_),
03908     current1_(other.current1_),
03909     begin2_(other.begin2_),
03910     end2_(other.end2_),
03911     current2_(other.current2_),
03912     begin3_(other.begin3_),
03913     end3_(other.end3_),
03914     current3_(other.current3_),
03915     begin4_(other.begin4_),
03916     end4_(other.end4_),
03917     current4_(other.current4_) {
03918     ComputeCurrentValue();
03919 }
03920
03921     void ComputeCurrentValue() {
03922     if (!AtEnd())
03923         current_value_.reset(new ParamType(*current1_, *current2_, *current3_,
03924             *current4_));
03925     }
03926     bool AtEnd() const {
03927     // We must report iterator past the end of the range when either of the
03928     // component iterators has reached the end of its range.
03929     return
03930         current1_ == endl_ ||
03931         current2_ == end2_ ||
03932         current3_ == end3_ ||
03933         current4_ == end4_;
03934     }
03935
03936     // No implementation - assignment is unsupported.
03937     void operator=(const Iterator& other);
03938
03939     const ParamGeneratorInterface<ParamType>* const base_;
03940     // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
03941     // current[i]_ is the actual traversing iterator.
03942     const typename ParamGenerator<T1>::iterator begin1_;
03943     const typename ParamGenerator<T1>::iterator endl_;
03944     typename ParamGenerator<T1>::iterator current1_;
03945     const typename ParamGenerator<T2>::iterator begin2_;
03946     const typename ParamGenerator<T2>::iterator end2_;
03947     typename ParamGenerator<T2>::iterator current2_;
03948     const typename ParamGenerator<T3>::iterator begin3_;
03949     const typename ParamGenerator<T3>::iterator end3_;
03950     typename ParamGenerator<T3>::iterator current3_;
03951     const typename ParamGenerator<T4>::iterator begin4_;
03952     const typename ParamGenerator<T4>::iterator end4_;
03953     typename ParamGenerator<T4>::iterator current4_;
03954     linked_ptr<ParamType> current_value_;
03955 }; // class CartesianProductGenerator4::Iterator
03956
03957     // No implementation - assignment is unsupported.
03958     void operator=(const CartesianProductGenerator4& other);
03959
03960     const ParamGenerator<T1> g1_;
03961     const ParamGenerator<T2> g2_;
03962     const ParamGenerator<T3> g3_;
03963     const ParamGenerator<T4> g4_;
03964 }; // class CartesianProductGenerator4
03965
03966 template <typename T1, typename T2, typename T3, typename T4, typename T5>
03967 class CartesianProductGenerator5
03968     : public ParamGeneratorInterface<::testing::tuple<T1, T2, T3, T4, T5> > {
03969     public:
03970     typedef ::testing::tuple<T1, T2, T3, T4, T5> ParamType;
03971     CartesianProductGenerator5(const ParamGenerator<T1>& g1,
03972         const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3,
03973         const ParamGenerator<T4>& g4, const ParamGenerator<T5>& g5)
03974         : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5) {}
03975     virtual ~CartesianProductGenerator5() {}

```

```

03978
03979     virtual ParamIteratorInterface<ParamType>* Begin() const {
03980         return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin(), g3_,
03981             g3_.begin(), g4_, g4_.begin(), g5_, g5_.begin());
03982     }
03983     virtual ParamIteratorInterface<ParamType>* End() const {
03984         return new Iterator(this, g1_, g1_.end(), g2_, g2_.end(), g3_,
03985             g3_.end(), g4_, g4_.end(), g5_, g5_.end());
03986     }
03987
03988 private:
03989     class Iterator : public ParamIteratorInterface<ParamType> {
03990     public:
03991         Iterator(const ParamGeneratorInterface<ParamType>* base,
03992             const ParamGenerator<T1>& g1,
03993             const typename ParamGenerator<T1>::iterator& current1,
03994             const ParamGenerator<T2>& g2,
03995             const typename ParamGenerator<T2>::iterator& current2,
03996             const ParamGenerator<T3>& g3,
03997             const typename ParamGenerator<T3>::iterator& current3,
03998             const ParamGenerator<T4>& g4,
03999             const typename ParamGenerator<T4>::iterator& current4,
04000             const ParamGenerator<T5>& g5,
04001             const typename ParamGenerator<T5>::iterator& current5)
04002             : base_(base),
04003                 begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
04004                 begin2_(g2.begin()), end2_(g2.end()), current2_(current2),
04005                 begin3_(g3.begin()), end3_(g3.end()), current3_(current3),
04006                 begin4_(g4.begin()), end4_(g4.end()), current4_(current4),
04007                 begin5_(g5.begin()), end5_(g5.end()), current5_(current5) {
04008             ComputeCurrentValue();
04009         }
04010     ~Iterator() {}
04011
04012     virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
04013         return base_;
04014     }
04015     // Advance should not be called on beyond-of-range iterators
04016     // so no component iterators must be beyond end of range, either.
04017     virtual void Advance() {
04018         assert(!AtEnd());
04019         ++current5_;
04020         if (current5_ == end5_) {
04021             current5_ = begin5_;
04022             ++current4_;
04023         }
04024         if (current4_ == end4_) {
04025             current4_ = begin4_;
04026             ++current3_;
04027         }
04028         if (current3_ == end3_) {
04029             current3_ = begin3_;
04030             ++current2_;
04031         }
04032         if (current2_ == end2_) {
04033             current2_ = begin2_;
04034             ++current1_;
04035         }
04036         ComputeCurrentValue();
04037     }
04038     virtual ParamIteratorInterface<ParamType>* Clone() const {
04039         return new Iterator(*this);
04040     }
04041     virtual const ParamType* Current() const { return current_value_.get(); }
04042     virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
04043         // Having the same base generator guarantees that the other
04044         // iterator is of the same type and we can downcast.
04045         GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
04046             << "The program attempted to compare iterators "
04047             << "from different generators." << std::endl;
04048         const Iterator* typed_other =
04049             CheckedDowncastToActualType<const Iterator>(&other);
04050         // We must report iterators equal if they both point beyond their
04051         // respective ranges. That can happen in a variety of fashions,
04052         // so we have to consult AtEnd().
04053         return (AtEnd() && typed_other->AtEnd()) ||
04054             (
04055                 current1_ == typed_other->current1_ &&
04056                 current2_ == typed_other->current2_ &&
04057                 current3_ == typed_other->current3_ &&
04058                 current4_ == typed_other->current4_ &&
04059                 current5_ == typed_other->current5_);
04060     }
04061
04062 private:
04063     Iterator(const Iterator& other)
04064         : base_(other.base_),

```

```

04065     begin1_(other.begin1_),
04066     endl_(other.end1_),
04067     current1_(other.current1_),
04068     begin2_(other.begin2_),
04069     end2_(other.end2_),
04070     current2_(other.current2_),
04071     begin3_(other.begin3_),
04072     end3_(other.end3_),
04073     current3_(other.current3_),
04074     begin4_(other.begin4_),
04075     end4_(other.end4_),
04076     current4_(other.current4_),
04077     begin5_(other.begin5_),
04078     end5_(other.end5_),
04079     current5_(other.current5_) {
04080     ComputeCurrentValue();
04081 }
04082
04083 void ComputeCurrentValue() {
04084     if (!AtEnd())
04085         current_value_.reset(new ParamType(*current1_, *current2_, *current3_,
04086                         *current4_, *current5_));
04087 }
04088 bool AtEnd() const {
04089     // We must report iterator past the end of the range when either of the
04090     // component iterators has reached the end of its range.
04091     return
04092         current1_ == end1_ ||
04093         current2_ == end2_ ||
04094         current3_ == end3_ ||
04095         current4_ == end4_ ||
04096         current5_ == end5_;
04097 }
04098
04099 // No implementation - assignment is unsupported.
04100 void operator=(const Iterator& other);
04101
04102 const ParamGeneratorInterface<ParamType>* const base_=
04103 // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
04104 // current[i]_ is the actual traversing iterator.
04105 const typename ParamGenerator<T1>::iterator begin1_;
04106 const typename ParamGenerator<T1>::iterator endl_;
04107 typename ParamGenerator<T1>::iterator current1_;
04108 const typename ParamGenerator<T2>::iterator begin2_;
04109 const typename ParamGenerator<T2>::iterator endl2_;
04110 typename ParamGenerator<T2>::iterator current2_;
04111 const typename ParamGenerator<T3>::iterator begin3_;
04112 const typename ParamGenerator<T3>::iterator endl3_;
04113 typename ParamGenerator<T3>::iterator current3_;
04114 const typename ParamGenerator<T4>::iterator begin4_;
04115 const typename ParamGenerator<T4>::iterator endl4_;
04116 typename ParamGenerator<T4>::iterator current4_;
04117 const typename ParamGenerator<T5>::iterator begin5_;
04118 const typename ParamGenerator<T5>::iterator endl5_;
04119 typename ParamGenerator<T5>::iterator current5_;
04120     linked_ptr<ParamType> current_value_;
04121 }; // class CartesianProductGenerator5::Iterator
04122
04123 // No implementation - assignment is unsupported.
04124 void operator=(const CartesianProductGenerator5& other);
04125
04126 const ParamGenerator<T1> g1_;
04127 const ParamGenerator<T2> g2_;
04128 const ParamGenerator<T3> g3_;
04129 const ParamGenerator<T4> g4_;
04130 const ParamGenerator<T5> g5_;
04131 }; // class CartesianProductGenerator5
04132
04133
04134 template <typename T1, typename T2, typename T3, typename T4, typename T5,
04135     typename T6>
04136 class CartesianProductGenerator6
04137     : public ParamGeneratorInterface<::testing::tuple<T1, T2, T3, T4, T5,
04138             T6> > {
04139 public:
04140     typedef ::testing::tuple<T1, T2, T3, T4, T5, T6> ParamType;
04141
04142     CartesianProductGenerator6(const ParamGenerator<T1>& g1,
04143         const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3,
04144         const ParamGenerator<T4>& g4, const ParamGenerator<T5>& g5,
04145         const ParamGenerator<T6>& g6)
04146         : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6) {}
04147     virtual ~CartesianProductGenerator6() {}
04148
04149     virtual ParamIteratorInterface<ParamType>* Begin() const {
04150         return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin(), g3_,
04151                         g3_.begin(), g4_, g4_.begin(), g5_, g5_.begin(), g6_, g6_.begin());
04151     }

```

```

04152     }
04153     virtual ParamIteratorInterface<ParamType>* End() const {
04154         return new Iterator(this, g1_, g1_.end(), g2_, g2_.end(), g3_, g3_.end(),
04155             g4_, g4_.end(), g5_, g5_.end(), g6_, g6_.end());
04156     }
04157
04158     private:
04159     class Iterator : public ParamIteratorInterface<ParamType> {
04160     public:
04161         Iterator(const ParamGeneratorInterface<ParamType>* base,
04162             const ParamGenerator<T1>& g1,
04163             const typename ParamGenerator<T1>::iterator& current1,
04164             const ParamGenerator<T2>& g2,
04165             const typename ParamGenerator<T2>::iterator& current2,
04166             const ParamGenerator<T3>& g3,
04167             const typename ParamGenerator<T3>::iterator& current3,
04168             const ParamGenerator<T4>& g4,
04169             const typename ParamGenerator<T4>::iterator& current4,
04170             const ParamGenerator<T5>& g5,
04171             const typename ParamGenerator<T5>::iterator& current5,
04172             const ParamGenerator<T6>& g6,
04173             const typename ParamGenerator<T6>::iterator& current6)
04174         : base_(base),
04175             begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
04176             begin2_(g2.begin()), end2_(g2.end()), current2_(current2),
04177             begin3_(g3.begin()), end3_(g3.end()), current3_(current3),
04178             begin4_(g4.begin()), end4_(g4.end()), current4_(current4),
04179             begin5_(g5.begin()), end5_(g5.end()), current5_(current5),
04180             begin6_(g6.begin()), end6_(g6.end()), current6_(current6) {
04181             ComputeCurrentValue();
04182         }
04183     virtual ~Iterator() {}
04184
04185     virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
04186         return base_;
04187     }
04188     // Advance should not be called on beyond-of-range iterators
04189     // so no component iterators must be beyond end of range, either.
04190     virtual void Advance() {
04191         assert(!AtEnd());
04192         ++current6_;
04193         if (current6_ == end6_) {
04194             current6_ = begin6_;
04195             ++current5_;
04196         }
04197         if (current5_ == end5_) {
04198             current5_ = begin5_;
04199             ++current4_;
04200         }
04201         if (current4_ == end4_) {
04202             current4_ = begin4_;
04203             ++current3_;
04204         }
04205         if (current3_ == end3_) {
04206             current3_ = begin3_;
04207             ++current2_;
04208         }
04209         if (current2_ == end2_) {
04210             current2_ = begin2_;
04211             ++current1_;
04212         }
04213         ComputeCurrentValue();
04214     }
04215     virtual ParamIteratorInterface<ParamType>* Clone() const {
04216         return new Iterator(*this);
04217     }
04218     virtual const ParamType* Current() const { return current_value_.get(); }
04219     virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
04220         // Having the same base generator guarantees that the other
04221         // iterator is of the same type and we can downcast.
04222         GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
04223             << "The program attempted to compare iterators "
04224             << "from different generators." << std::endl;
04225         const Iterator* typed_other =
04226             CheckedDowncastToActualType<const Iterator>(&other);
04227         // We must report iterators equal if they both point beyond their
04228         // respective ranges. That can happen in a variety of fashions,
04229         // so we have to consult AtEnd().
04230         return (AtEnd() && typed_other->AtEnd()) ||
04231             (
04232                 current1_ == typed_other->current1_ &&
04233                 current2_ == typed_other->current2_ &&
04234                 current3_ == typed_other->current3_ &&
04235                 current4_ == typed_other->current4_ &&
04236                 current5_ == typed_other->current5_ &&
04237                 current6_ == typed_other->current6_);
04238     }

```

```

04239
04240     private:
04241         Iterator(const Iterator& other)
04242             : base_(other.base_),
04243                 begin1_(other.begin1_),
04244                 endl_(other.end1_),
04245                 current1_(other.current1_),
04246                 begin2_(other.begin2_),
04247                 end2_(other.end2_),
04248                 current2_(other.current2_),
04249                 begin3_(other.begin3_),
04250                 end3_(other.end3_),
04251                 current3_(other.current3_),
04252                 begin4_(other.begin4_),
04253                 end4_(other.end4_),
04254                 current4_(other.current4_),
04255                 begin5_(other.begin5_),
04256                 end5_(other.end5_),
04257                 current5_(other.current5_),
04258                 begin6_(other.begin6_),
04259                 end6_(other.end6_),
04260                 current6_(other.current6_) {
04261         ComputeCurrentValue();
04262     }
04263
04264     void ComputeCurrentValue() {
04265         if (!AtEnd())
04266             current_value_.reset(new ParamType(*current1_, *current2_, *current3_,
04267                                     *current4_, *current5_, *current6_));
04268     }
04269     bool AtEnd() const {
04270         // We must report iterator past the end of the range when either of the
04271         // component iterators has reached the end of its range.
04272         return
04273             current1_ == endl_ ||
04274             current2_ == end2_ ||
04275             current3_ == end3_ ||
04276             current4_ == end4_ ||
04277             current5_ == end5_ ||
04278             current6_ == end6_;
04279     }
04280
04281     // No implementation - assignment is unsupported.
04282     void operator=(const Iterator& other);
04283
04284     const ParamGeneratorInterface<ParamType>* const base_;
04285     // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
04286     // current[i]_ is the actual traversing iterator.
04287     const typename ParamGenerator<T1>::iterator begin1_;
04288     const typename ParamGenerator<T1>::iterator endl_;
04289     typename ParamGenerator<T1>::iterator current1_;
04290     const typename ParamGenerator<T2>::iterator begin2_;
04291     const typename ParamGenerator<T2>::iterator end2_;
04292     typename ParamGenerator<T2>::iterator current2_;
04293     const typename ParamGenerator<T3>::iterator begin3_;
04294     const typename ParamGenerator<T3>::iterator end3_;
04295     typename ParamGenerator<T3>::iterator current3_;
04296     const typename ParamGenerator<T4>::iterator begin4_;
04297     const typename ParamGenerator<T4>::iterator end4_;
04298     typename ParamGenerator<T4>::iterator current4_;
04299     const typename ParamGenerator<T5>::iterator begin5_;
04300     const typename ParamGenerator<T5>::iterator end5_;
04301     typename ParamGenerator<T5>::iterator current5_;
04302     const typename ParamGenerator<T6>::iterator begin6_;
04303     const typename ParamGenerator<T6>::iterator end6_;
04304     typename ParamGenerator<T6>::iterator current6_;
04305     linked_ptr<ParamType> current_value_;
04306 }; // class CartesianProductGenerator6::Iterator
04307
04308     // No implementation - assignment is unsupported.
04309     void operator=(const CartesianProductGenerator6& other);
04310
04311     const ParamGenerator<T1> g1_;
04312     const ParamGenerator<T2> g2_;
04313     const ParamGenerator<T3> g3_;
04314     const ParamGenerator<T4> g4_;
04315     const ParamGenerator<T5> g5_;
04316     const ParamGenerator<T6> g6_;
04317 }; // class CartesianProductGenerator6
04318
04319
04320     template <typename T1, typename T2, typename T3, typename T4, typename T5,
04321             typename T6, typename T7>
04322     class CartesianProductGenerator7
04323         : public ParamGeneratorInterface<::testing::tuple<T1, T2, T3, T4, T5, T6,
04324                                         T7> > {
04325     public:

```

```

04326     typedef ::testing::tuple<T1, T2, T3, T4, T5, T6, T7> ParamType;
04327
04328     CartesianProductGenerator7(const ParamGenerator<T1>& g1,
04329         const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3,
04330         const ParamGenerator<T4>& g4, const ParamGenerator<T5>& g5,
04331         const ParamGenerator<T6>& g6, const ParamGenerator<T7>& g7)
04332         : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7) {}
04333     ~CartesianProductGenerator7() {}
04334
04335     virtual ParamIteratorInterface<ParamType>* Begin() const {
04336         return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin(), g3_,
04337             g3_.begin(), g4_, g4_.begin(), g5_, g5_.begin(), g6_, g6_.begin(), g7_,
04338             g7_.begin());
04339     }
04340     virtual ParamIteratorInterface<ParamType>* End() const {
04341         return new Iterator(this, g1_, g1_.end(), g2_, g2_.end(), g3_, g3_.end(),
04342             g4_, g4_.end(), g5_, g5_.end(), g6_, g6_.end(), g7_, g7_.end());
04343     }
04344
04345     private:
04346     class Iterator : public ParamIteratorInterface<ParamType> {
04347     public:
04348         Iterator(const ParamGeneratorInterface<ParamType>* base,
04349             const ParamGenerator<T1>& g1,
04350             const typename ParamGenerator<T1>::iterator& current1,
04351             const ParamGenerator<T2>& g2,
04352             const typename ParamGenerator<T2>::iterator& current2,
04353             const ParamGenerator<T3>& g3,
04354             const typename ParamGenerator<T3>::iterator& current3,
04355             const ParamGenerator<T4>& g4,
04356             const typename ParamGenerator<T4>::iterator& current4,
04357             const ParamGenerator<T5>& g5,
04358             const typename ParamGenerator<T5>::iterator& current5,
04359             const ParamGenerator<T6>& g6,
04360             const typename ParamGenerator<T6>::iterator& current6,
04361             const ParamGenerator<T7>& g7,
04362             const typename ParamGenerator<T7>::iterator& current7)
04363         : base_(base),
04364             begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
04365             begin2_(g2.begin()), end2_(g2.end()), current2_(current2),
04366             begin3_(g3.begin()), end3_(g3.end()), current3_(current3),
04367             begin4_(g4.begin()), end4_(g4.end()), current4_(current4),
04368             begin5_(g5.begin()), end5_(g5.end()), current5_(current5),
04369             begin6_(g6.begin()), end6_(g6.end()), current6_(current6),
04370             begin7_(g7.begin()), end7_(g7.end()), current7_(current7) {
04371             ComputeCurrentValue();
04372         }
04373     virtual ~Iterator() {}
04374
04375     virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
04376         return base_;
04377     }
04378     // Advance should not be called on beyond-of-range iterators
04379     // so no component iterators must be beyond end of range, either.
04380     virtual void Advance() {
04381         assert(!AtEnd());
04382         ++current7_;
04383         if (current7_ == end7_) {
04384             current7_ = begin7_;
04385             ++current6_;
04386         }
04387         if (current6_ == end6_) {
04388             current6_ = begin6_;
04389             ++current5_;
04390         }
04391         if (current5_ == end5_) {
04392             current5_ = begin5_;
04393             ++current4_;
04394         }
04395         if (current4_ == end4_) {
04396             current4_ = begin4_;
04397             ++current3_;
04398         }
04399         if (current3_ == end3_) {
04400             current3_ = begin3_;
04401             ++current2_;
04402         }
04403         if (current2_ == end2_) {
04404             current2_ = begin2_;
04405             ++current1_;
04406         }
04407         ComputeCurrentValue();
04408     }
04409     virtual ParamIteratorInterface<ParamType>* Clone() const {
04410         return new Iterator(*this);
04411     }
04412     virtual const ParamType* Current() const { return current_value_.get(); }

```

```

04413     virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
04414         // Having the same base generator guarantees that the other
04415         // iterator is of the same type and we can downcast.
04416         GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
04417             << "The program attempted to compare iterators "
04418             << "from different generators." << std::endl;
04419         const Iterator* typed_other =
04420             CheckedDowncastToActualType<const Iterator>(&other);
04421         // We must report iterators equal if they both point beyond their
04422         // respective ranges. That can happen in a variety of fashions,
04423         // so we have to consult AtEnd().
04424         return (AtEnd() && typed_other->AtEnd()) ||
04425             (
04426                 current1_ == typed_other->current1_ &&
04427                 current2_ == typed_other->current2_ &&
04428                 current3_ == typed_other->current3_ &&
04429                 current4_ == typed_other->current4_ &&
04430                 current5_ == typed_other->current5_ &&
04431                 current6_ == typed_other->current6_ &&
04432                 current7_ == typed_other->current7_);
04433     }
04434
04435     private:
04436     Iterator(const Iterator& other)
04437         : base_(other.base_),
04438         begin1_(other.begin1_),
04439         end1_(other.end1_),
04440         current1_(other.current1_),
04441         begin2_(other.begin2_),
04442         end2_(other.end2_),
04443         current2_(other.current2_),
04444         begin3_(other.begin3_),
04445         end3_(other.end3_),
04446         current3_(other.current3_),
04447         begin4_(other.begin4_),
04448         end4_(other.end4_),
04449         current4_(other.current4_),
04450         begin5_(other.begin5_),
04451         end5_(other.end5_),
04452         current5_(other.current5_),
04453         begin6_(other.begin6_),
04454         end6_(other.end6_),
04455         current6_(other.current6_),
04456         begin7_(other.begin7_),
04457         end7_(other.end7_),
04458         current7_(other.current7_) {
04459         ComputeCurrentValue();
04460     }
04461
04462     void ComputeCurrentValue() {
04463         if (!AtEnd())
04464             current_value_.reset(new ParamType(*current1_, *current2_, *current3_,
04465                         *current4_, *current5_, *current6_, *current7_));
04466     }
04467     bool AtEnd() const {
04468         // We must report iterator past the end of the range when either of the
04469         // component iterators has reached the end of its range.
04470         return
04471             current1_ == end1_ ||
04472             current2_ == end2_ ||
04473             current3_ == end3_ ||
04474             current4_ == end4_ ||
04475             current5_ == end5_ ||
04476             current6_ == end6_ ||
04477             current7_ == end7_;
04478     }
04479
04480     // No implementation - assignment is unsupported.
04481     void operator=(const Iterator& other);
04482
04483     const ParamGeneratorInterface<ParamType>*& const base_>;
04484     // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
04485     // current[i]_ is the actual traversing iterator.
04486     const typename ParamGenerator<T1>::iterator begin1_;
04487     const typename ParamGenerator<T1>::iterator end1_;
04488     typename ParamGenerator<T1>::iterator current1_;
04489     const typename ParamGenerator<T2>::iterator begin2_;
04490     const typename ParamGenerator<T2>::iterator end2_;
04491     typename ParamGenerator<T2>::iterator current2_;
04492     const typename ParamGenerator<T3>::iterator begin3_;
04493     const typename ParamGenerator<T3>::iterator end3_;
04494     typename ParamGenerator<T3>::iterator current3_;
04495     const typename ParamGenerator<T4>::iterator begin4_;
04496     const typename ParamGenerator<T4>::iterator end4_;
04497     typename ParamGenerator<T4>::iterator current4_;
04498     const typename ParamGenerator<T5>::iterator begin5_;
04499     const typename ParamGenerator<T5>::iterator end5_;

```

```

04500     typename ParamGenerator<T5>::iterator current5_;
04501     const typename ParamGenerator<T6>::iterator begin6_;
04502     const typename ParamGenerator<T6>::iterator end6_;
04503     typename ParamGenerator<T6>::iterator current6_;
04504     const typename ParamGenerator<T7>::iterator begin7_;
04505     const typename ParamGenerator<T7>::iterator end7_;
04506     typename ParamGenerator<T7>::iterator current7_;
04507     linked_ptr<ParamType> current_value_;
04508 }; // class CartesianProductGenerator7::Iterator
04509
04510 // No implementation - assignment is unsupported.
04511 void operator=(const CartesianProductGenerator7& other);
04512
04513 const ParamGenerator<T1> g1_;
04514 const ParamGenerator<T2> g2_;
04515 const ParamGenerator<T3> g3_;
04516 const ParamGenerator<T4> g4_;
04517 const ParamGenerator<T5> g5_;
04518 const ParamGenerator<T6> g6_;
04519 const ParamGenerator<T7> g7_;
04520 }; // class CartesianProductGenerator7
04521
04522
04523 template <typename T1, typename T2, typename T3, typename T4, typename T5,
04524     typename T6, typename T7, typename T8>
04525 class CartesianProductGenerator8
04526     : public ParamGeneratorInterface<::testing::tuple<T1, T2, T3, T4, T5, T6,
04527         T7, T8> > {
04528 public:
04529     typedef ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8> ParamType;
04530
04531     CartesianProductGenerator8(const ParamGenerator<T1>& g1,
04532         const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3,
04533         const ParamGenerator<T4>& g4, const ParamGenerator<T5>& g5,
04534         const ParamGenerator<T6>& g6, const ParamGenerator<T7>& g7,
04535         const ParamGenerator<T8>& g8)
04536         : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7),
04537         g8_(g8) {}
04538     virtual ~CartesianProductGenerator8() {}
04539
04540     virtual ParamIteratorInterface<ParamType>* Begin() const {
04541         return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin(), g3_,
04542             g3_.begin(), g4_, g4_.begin(), g5_, g5_.begin(), g6_, g6_.begin(), g7_,
04543             g7_.begin(), g8_, g8_.begin());
04544     }
04545     virtual ParamIteratorInterface<ParamType>* End() const {
04546         return new Iterator(this, g1_, g1_.end(), g2_, g2_.end(), g3_, g3_.end(),
04547             g4_, g4_.end(), g5_, g5_.end(), g6_, g6_.end(), g7_, g7_.end(), g8_,
04548             g8_.end());
04549     }
04550
04551 private:
04552     class Iterator : public ParamIteratorInterface<ParamType> {
04553     public:
04554         Iterator(const ParamGeneratorInterface<ParamType>* base,
04555             const ParamGenerator<T1>& g1,
04556             const typename ParamGenerator<T1>::iterator& current1,
04557             const ParamGenerator<T2>& g2,
04558             const typename ParamGenerator<T2>::iterator& current2,
04559             const ParamGenerator<T3>& g3,
04560             const typename ParamGenerator<T3>::iterator& current3,
04561             const ParamGenerator<T4>& g4,
04562             const typename ParamGenerator<T4>::iterator& current4,
04563             const ParamGenerator<T5>& g5,
04564             const typename ParamGenerator<T5>::iterator& current5,
04565             const ParamGenerator<T6>& g6,
04566             const typename ParamGenerator<T6>::iterator& current6,
04567             const ParamGenerator<T7>& g7,
04568             const typename ParamGenerator<T7>::iterator& current7,
04569             const ParamGenerator<T8>& g8,
04570             const typename ParamGenerator<T8>::iterator& current8)
04571             : base_(base),
04572                 begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
04573                 begin2_(g2.begin()), end2_(g2.end()), current2_(current2),
04574                 begin3_(g3.begin()), end3_(g3.end()), current3_(current3),
04575                 begin4_(g4.begin()), end4_(g4.end()), current4_(current4),
04576                 begin5_(g5.begin()), end5_(g5.end()), current5_(current5),
04577                 begin6_(g6.begin()), end6_(g6.end()), current6_(current6),
04578                 begin7_(g7.begin()), end7_(g7.end()), current7_(current7),
04579                 begin8_(g8.begin()), end8_(g8.end()), current8_(current8) {
04580             ComputeCurrentValue();
04581         }
04582         virtual ~Iterator() {}
04583
04584         virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
04585             return base_;
04586         }

```

```

04587 // Advance should not be called on beyond-of-range iterators
04588 // so no component iterators must be beyond end of range, either.
04589 virtual void Advance() {
04590     assert(!AtEnd());
04591     ++current8_;
04592     if (current8_ == end8_) {
04593         current8_ = begin8_;
04594         ++current7_;
04595     }
04596     if (current7_ == end7_) {
04597         current7_ = begin7_;
04598         ++current6_;
04599     }
04600     if (current6_ == end6_) {
04601         current6_ = begin6_;
04602         ++current5_;
04603     }
04604     if (current5_ == end5_) {
04605         current5_ = begin5_;
04606         ++current4_;
04607     }
04608     if (current4_ == end4_) {
04609         current4_ = begin4_;
04610         ++current3_;
04611     }
04612     if (current3_ == end3_) {
04613         current3_ = begin3_;
04614         ++current2_;
04615     }
04616     if (current2_ == end2_) {
04617         current2_ = begin2_;
04618         ++current1_;
04619     }
04620     ComputeCurrentValue();
04621 }
04622 virtual ParamIteratorInterface<ParamType>* Clone() const {
04623     return new Iterator(*this);
04624 }
04625 virtual const ParamType* Current() const { return current_value_.get(); }
04626 virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
04627     // Having the same base generator guarantees that the other
04628     // iterator is of the same type and we can downcast.
04629     GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
04630         << "The program attempted to compare iterators "
04631         << "from different generators." << std::endl;
04632     const Iterator* typed_other =
04633         CheckedDowncastToActualType<const Iterator>(&other);
04634     // We must report iterators equal if they both point beyond their
04635     // respective ranges. That can happen in a variety of fashions,
04636     // so we have to consult AtEnd().
04637     return (AtEnd() && typed_other->AtEnd()) ||
04638         (
04639             current1_ == typed_other->current1_ &&
04640             current2_ == typed_other->current2_ &&
04641             current3_ == typed_other->current3_ &&
04642             current4_ == typed_other->current4_ &&
04643             current5_ == typed_other->current5_ &&
04644             current6_ == typed_other->current6_ &&
04645             current7_ == typed_other->current7_ &&
04646             current8_ == typed_other->current8_);
04647 }
04648
04649 private:
04650     Iterator(const Iterator& other)
04651         : base_(other.base_),
04652         begin1_(other.begin1_),
04653         end1_(other.end1_),
04654         current1_(other.current1_),
04655         begin2_(other.begin2_),
04656         end2_(other.end2_),
04657         current2_(other.current2_),
04658         begin3_(other.begin3_),
04659         end3_(other.end3_),
04660         current3_(other.current3_),
04661         begin4_(other.begin4_),
04662         end4_(other.end4_),
04663         current4_(other.current4_),
04664         begin5_(other.begin5_),
04665         end5_(other.end5_),
04666         current5_(other.current5_),
04667         begin6_(other.begin6_),
04668         end6_(other.end6_),
04669         current6_(other.current6_),
04670         begin7_(other.begin7_),
04671         end7_(other.end7_),
04672         current7_(other.current7_),
04673         begin8_(other.begin8_),

```

```

04674     end8_(other.end8_),
04675     current8_(other.current8_) {
04676     ComputeCurrentValue();
04677 }
04678
04679     void ComputeCurrentValue() {
04680     if (!AtEnd())
04681         current_value_.reset(new ParamType(*current1_, *current2_, *current3_,
04682                         *current4_, *current5_, *current6_, *current7_, *current8_));
04683 }
04684     bool AtEnd() const {
04685     // We must report iterator past the end of the range when either of the
04686     // component iterators has reached the end of its range.
04687     return
04688         current1_ == end1_ ||
04689         current2_ == end2_ ||
04690         current3_ == end3_ ||
04691         current4_ == end4_ ||
04692         current5_ == end5_ ||
04693         current6_ == end6_ ||
04694         current7_ == end7_ ||
04695         current8_ == end8_;
04696 }
04697
04698 // No implementation - assignment is unsupported.
04699 void operator=(const Iterator& other);
04700
04701     const ParamGeneratorInterface<ParamType>* const base_;
04702 // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
04703 // current[i]_ is the actual traversing iterator.
04704     const typename ParamGenerator<T1>::iterator begin1_;
04705     const typename ParamGenerator<T1>::iterator end1_;
04706     typename ParamGenerator<T1>::iterator current1_;
04707     const typename ParamGenerator<T2>::iterator begin2_;
04708     const typename ParamGenerator<T2>::iterator end2_;
04709     typename ParamGenerator<T2>::iterator current2_;
04710     const typename ParamGenerator<T3>::iterator begin3_;
04711     const typename ParamGenerator<T3>::iterator end3_;
04712     typename ParamGenerator<T3>::iterator current3_;
04713     const typename ParamGenerator<T4>::iterator begin4_;
04714     const typename ParamGenerator<T4>::iterator end4_;
04715     typename ParamGenerator<T4>::iterator current4_;
04716     const typename ParamGenerator<T5>::iterator begin5_;
04717     const typename ParamGenerator<T5>::iterator end5_;
04718     typename ParamGenerator<T5>::iterator current5_;
04719     const typename ParamGenerator<T6>::iterator begin6_;
04720     const typename ParamGenerator<T6>::iterator end6_;
04721     typename ParamGenerator<T6>::iterator current6_;
04722     const typename ParamGenerator<T7>::iterator begin7_;
04723     const typename ParamGenerator<T7>::iterator end7_;
04724     typename ParamGenerator<T7>::iterator current7_;
04725     const typename ParamGenerator<T8>::iterator begin8_;
04726     const typename ParamGenerator<T8>::iterator end8_;
04727     typename ParamGenerator<T8>::iterator current8_;
04728     linked_ptr<ParamType> current_value_;
04729 }; // class CartesianProductGenerator8::Iterator
04730
04731 // No implementation - assignment is unsupported.
04732 void operator=(const CartesianProductGenerator8& other);
04733
04734     const ParamGenerator<T1> g1_;
04735     const ParamGenerator<T2> g2_;
04736     const ParamGenerator<T3> g3_;
04737     const ParamGenerator<T4> g4_;
04738     const ParamGenerator<T5> g5_;
04739     const ParamGenerator<T6> g6_;
04740     const ParamGenerator<T7> g7_;
04741     const ParamGenerator<T8> g8_;
04742 }; // class CartesianProductGenerator8
04743
04744
04745 template <typename T1, typename T2, typename T3, typename T4, typename T5,
04746           typename T6, typename T7, typename T8, typename T9>
04747 class CartesianProductGenerator9
04748 : public ParamGeneratorInterface< ::testing::tuple<T1, T2, T3, T4, T5, T6,
04749           T7, T8, T9> > {
04750 public:
04751     typedef ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8, T9> ParamType;
04752
04753     CartesianProductGenerator9(const ParamGenerator<T1>& g1,
04754         const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3,
04755         const ParamGenerator<T4>& g4, const ParamGenerator<T5>& g5,
04756         const ParamGenerator<T6>& g6, const ParamGenerator<T7>& g7,
04757         const ParamGenerator<T8>& g8, const ParamGenerator<T9>& g9)
04758         : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7), g8_(g8),
04759         g9_(g9) {}
04760     virtual ~CartesianProductGenerator9() {}

```

```

04761
04762     virtual ParamIteratorInterface<ParamType>* Begin() const {
04763         return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin(), g3_,
04764             g3_.begin(), g4_, g4_.begin(), g5_, g5_.begin(), g6_, g6_.begin(), g7_,
04765             g7_.begin(), g8_, g8_.begin(), g9_, g9_.begin());
04766     }
04767     virtual ParamIteratorInterface<ParamType>* End() const {
04768         return new Iterator(this, g1_, g1_.end(), g2_, g2_.end(), g3_, g3_.end(),
04769             g4_, g4_.end(), g5_, g5_.end(), g6_, g6_.end(), g7_, g7_.end(), g8_,
04770             g8_.end(), g9_, g9_.end());
04771     }
04772
04773 private:
04774     class Iterator : public ParamIteratorInterface<ParamType> {
04775     public:
04776         Iterator(const ParamGeneratorInterface<ParamType>* base,
04777             const ParamGenerator<T1>& g1,
04778             const typename ParamGenerator<T1>::iterator& current1,
04779             const ParamGenerator<T2>& g2,
04780             const typename ParamGenerator<T2>::iterator& current2,
04781             const ParamGenerator<T3>& g3,
04782             const typename ParamGenerator<T3>::iterator& current3,
04783             const ParamGenerator<T4>& g4,
04784             const typename ParamGenerator<T4>::iterator& current4,
04785             const ParamGenerator<T5>& g5,
04786             const typename ParamGenerator<T5>::iterator& current5,
04787             const ParamGenerator<T6>& g6,
04788             const typename ParamGenerator<T6>::iterator& current6,
04789             const ParamGenerator<T7>& g7,
04790             const typename ParamGenerator<T7>::iterator& current7,
04791             const ParamGenerator<T8>& g8,
04792             const typename ParamGenerator<T8>::iterator& current8,
04793             const ParamGenerator<T9>& g9,
04794             const typename ParamGenerator<T9>::iterator& current9)
04795             : base_(base),
04796                 begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
04797                 begin2_(g2.begin()), end2_(g2.end()), current2_(current2),
04798                 begin3_(g3.begin()), end3_(g3.end()), current3_(current3),
04799                 begin4_(g4.begin()), end4_(g4.end()), current4_(current4),
04800                 begin5_(g5.begin()), end5_(g5.end()), current5_(current5),
04801                 begin6_(g6.begin()), end6_(g6.end()), current6_(current6),
04802                 begin7_(g7.begin()), end7_(g7.end()), current7_(current7),
04803                 begin8_(g8.begin()), end8_(g8.end()), current8_(current8),
04804                 begin9_(g9.begin()), end9_(g9.end()), current9_(current9)    {
04805         ComputeCurrentValue();
04806     }
04807     virtual ~Iterator() {}
04808
04809     virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
04810         return base_;
04811     }
04812     // Advance should not be called on beyond-of-range iterators
04813     // so no component iterators must be beyond end of range, either.
04814     virtual void Advance() {
04815         assert(!AtEnd());
04816         ++current9_;
04817         if (current9_ == end9_) {
04818             current9_ = begin9_;
04819             ++current8_;
04820         }
04821         if (current8_ == end8_) {
04822             current8_ = begin8_;
04823             ++current7_;
04824         }
04825         if (current7_ == end7_) {
04826             current7_ = begin7_;
04827             ++current6_;
04828         }
04829         if (current6_ == end6_) {
04830             current6_ = begin6_;
04831             ++current5_;
04832         }
04833         if (current5_ == end5_) {
04834             current5_ = begin5_;
04835             ++current4_;
04836         }
04837         if (current4_ == end4_) {
04838             current4_ = begin4_;
04839             ++current3_;
04840         }
04841         if (current3_ == end3_) {
04842             current3_ = begin3_;
04843             ++current2_;
04844         }
04845         if (current2_ == end2_) {
04846             current2_ = begin2_;
04847             ++current1_;

```

```

04848     }
04849     ComputeCurrentValue();
04850 }
04851 virtual ParamIteratorInterface<ParamType>* Clone() const {
04852     return new Iterator(*this);
04853 }
04854 virtual const ParamType* Current() const { return current_value_.get(); }
04855 virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
04856     // Having the same base generator guarantees that the other
04857     // iterator is of the same type and we can downcast.
04858     GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
04859     « "The program attempted to compare iterators "
04860     « "from different generators." « std::endl;
04861     const Iterator* typed_other =
04862         CheckedDowncastToActualType<const Iterator>(&other);
04863     // We must report iterators equal if they both point beyond their
04864     // respective ranges. That can happen in a variety of fashions,
04865     // so we have to consult AtEnd().
04866     return (AtEnd() && typed_other->AtEnd()) ||
04867     (
04868         current1_ == typed_other->current1_ &&
04869         current2_ == typed_other->current2_ &&
04870         current3_ == typed_other->current3_ &&
04871         current4_ == typed_other->current4_ &&
04872         current5_ == typed_other->current5_ &&
04873         current6_ == typed_other->current6_ &&
04874         current7_ == typed_other->current7_ &&
04875         current8_ == typed_other->current8_ &&
04876         current9_ == typed_other->current9_);
04877 }
04878
04879 private:
04880     Iterator(const Iterator& other)
04881     : base_(other.base_),
04882     begin1_(other.begin1_),
04883     end1_(other.end1_),
04884     current1_(other.current1_),
04885     begin2_(other.begin2_),
04886     end2_(other.end2_),
04887     current2_(other.current2_),
04888     begin3_(other.begin3_),
04889     end3_(other.end3_),
04890     current3_(other.current3_),
04891     begin4_(other.begin4_),
04892     end4_(other.end4_),
04893     current4_(other.current4_),
04894     begin5_(other.begin5_),
04895     end5_(other.end5_),
04896     current5_(other.current5_),
04897     begin6_(other.begin6_),
04898     end6_(other.end6_),
04899     current6_(other.current6_),
04900     begin7_(other.begin7_),
04901     end7_(other.end7_),
04902     current7_(other.current7_),
04903     begin8_(other.begin8_),
04904     end8_(other.end8_),
04905     current8_(other.current8_),
04906     begin9_(other.begin9_),
04907     end9_(other.end9_),
04908     current9_(other.current9_) {
04909     ComputeCurrentValue();
04910 }
04911
04912 void ComputeCurrentValue() {
04913     if (!AtEnd())
04914         current_value_.reset(new ParamType(*current1_, *current2_, *current3_,
04915             *current4_, *current5_, *current6_, *current7_, *current8_,
04916             *current9_));
04917 }
04918 bool AtEnd() const {
04919     // We must report iterator past the end of the range when either of the
04920     // component iterators has reached the end of its range.
04921     return
04922         current1_ == end1_ ||
04923         current2_ == end2_ ||
04924         current3_ == end3_ ||
04925         current4_ == end4_ ||
04926         current5_ == end5_ ||
04927         current6_ == end6_ ||
04928         current7_ == end7_ ||
04929         current8_ == end8_ ||
04930         current9_ == end9_;
04931 }
04932
04933 // No implementation - assignment is unsupported.
04934 void operator=(const Iterator& other);

```

```

04935
04936     const ParamGeneratorInterface<ParamType>* const base_;
04937     // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
04938     // current[i]_ is the actual traversing iterator.
04939     const typename ParamGenerator<T1>::iterator begin1_;
04940     const typename ParamGenerator<T1>::iterator endl_;
04941     typename ParamGenerator<T1>::iterator current1_;
04942     const typename ParamGenerator<T2>::iterator begin2_;
04943     const typename ParamGenerator<T2>::iterator end2_;
04944     typename ParamGenerator<T2>::iterator current2_;
04945     const typename ParamGenerator<T3>::iterator begin3_;
04946     const typename ParamGenerator<T3>::iterator end3_;
04947     typename ParamGenerator<T3>::iterator current3_;
04948     const typename ParamGenerator<T4>::iterator begin4_;
04949     const typename ParamGenerator<T4>::iterator end4_;
04950     typename ParamGenerator<T4>::iterator current4_;
04951     const typename ParamGenerator<T5>::iterator begin5_;
04952     const typename ParamGenerator<T5>::iterator end5_;
04953     typename ParamGenerator<T5>::iterator current5_;
04954     const typename ParamGenerator<T6>::iterator begin6_;
04955     const typename ParamGenerator<T6>::iterator end6_;
04956     typename ParamGenerator<T6>::iterator current6_;
04957     const typename ParamGenerator<T7>::iterator begin7_;
04958     const typename ParamGenerator<T7>::iterator end7_;
04959     typename ParamGenerator<T7>::iterator current7_;
04960     const typename ParamGenerator<T8>::iterator begin8_;
04961     const typename ParamGenerator<T8>::iterator end8_;
04962     typename ParamGenerator<T8>::iterator current8_;
04963     const typename ParamGenerator<T9>::iterator begin9_;
04964     const typename ParamGenerator<T9>::iterator end9_;
04965     typename ParamGenerator<T9>::iterator current9_;
04966     linked_ptr<ParamType> current_value_;
04967 }; // class CartesianProductGenerator9::Iterator
04968
04969 // No implementation - assignment is unsupported.
04970 void operator=(const CartesianProductGenerator9& other);
04971
04972     const ParamGenerator<T1> g1_;
04973     const ParamGenerator<T2> g2_;
04974     const ParamGenerator<T3> g3_;
04975     const ParamGenerator<T4> g4_;
04976     const ParamGenerator<T5> g5_;
04977     const ParamGenerator<T6> g6_;
04978     const ParamGenerator<T7> g7_;
04979     const ParamGenerator<T8> g8_;
04980     const ParamGenerator<T9> g9_;
04981 }; // class CartesianProductGenerator9
04982
04983
04984 template <typename T1, typename T2, typename T3, typename T4, typename T5,
04985     typename T6, typename T7, typename T8, typename T9, typename T10>
04986 class CartesianProductGenerator10
04987     : public ParamGeneratorInterface<::testing::tuple<T1, T2, T3, T4, T5, T6,
04988         T7, T8, T9, T10> > {
04989 public:
04990     typedef ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10> ParamType;
04991
04992     CartesianProductGenerator10(const ParamGenerator<T1>& g1,
04993         const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3,
04994         const ParamGenerator<T4>& g4, const ParamGenerator<T5>& g5,
04995         const ParamGenerator<T6>& g6, const ParamGenerator<T7>& g7,
04996         const ParamGenerator<T8>& g8, const ParamGenerator<T9>& g9,
04997         const ParamGenerator<T10>& g10)
04998         : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7),
04999             g8_(g8), g9_(g9), g10_(g10) {}
05000     virtual ~CartesianProductGenerator10() {}
05001
05002     virtual ParamIteratorInterface<ParamType>* Begin() const {
05003         return new Iterator(this, g1_.begin(), g2_.begin(), g3_.begin(),
05004             g4_.begin(), g5_.begin(), g6_.begin(), g7_.begin(), g8_.begin(),
05005             g9_.begin(), g10_.begin());
05006     }
05007     virtual ParamIteratorInterface<ParamType>* End() const {
05008         return new Iterator(this, g1_.end(), g2_.end(), g3_.end(),
05009             g4_.end(), g5_.end(), g6_.end(), g7_.end(), g8_.end(),
05010             g9_.end(), g10_.end());
05011     }
05012
05013 private:
05014     class Iterator : public ParamIteratorInterface<ParamType> {
05015     public:
05016         Iterator(const ParamGeneratorInterface<ParamType>* base,
05017             const ParamGenerator<T1>& g1,
05018             const typename ParamGenerator<T1>::iterator& current1,
05019             const ParamGenerator<T2>& g2,
05020             const typename ParamGenerator<T2>::iterator& current2,
05021             const ParamGenerator<T3>& g3,

```

```

05022     const typename ParamGenerator<T3>::iterator& current3,
05023     const ParamGenerator<T4>& g4,
05024     const typename ParamGenerator<T4>::iterator& current4,
05025     const ParamGenerator<T5>& g5,
05026     const typename ParamGenerator<T5>::iterator& current5,
05027     const ParamGenerator<T6>& g6,
05028     const typename ParamGenerator<T6>::iterator& current6,
05029     const ParamGenerator<T7>& g7,
05030     const typename ParamGenerator<T7>::iterator& current7,
05031     const ParamGenerator<T8>& g8,
05032     const typename ParamGenerator<T8>::iterator& current8,
05033     const ParamGenerator<T9>& g9,
05034     const typename ParamGenerator<T9>::iterator& current9,
05035     const ParamGenerator<T10>& g10,
05036     const typename ParamGenerator<T10>::iterator& current10)
05037     : base_(base),
05038         begin1_(g1.begin()), endl_(g1.end()), current1_(current1),
05039         begin2_(g2.begin()), endl2_(g2.end()), current2_(current2),
05040         begin3_(g3.begin()), endl3_(g3.end()), current3_(current3),
05041         begin4_(g4.begin()), endl4_(g4.end()), current4_(current4),
05042         begin5_(g5.begin()), endl5_(g5.end()), current5_(current5),
05043         begin6_(g6.begin()), endl6_(g6.end()), current6_(current6),
05044         begin7_(g7.begin()), endl7_(g7.end()), current7_(current7),
05045         begin8_(g8.begin()), endl8_(g8.end()), current8_(current8),
05046         begin9_(g9.begin()), endl9_(g9.end()), current9_(current9),
05047         begin10_(g10.begin()), endl10_(g10.end()), current10_(current10)    {
05048     ComputeCurrentValue();
05049 }
05050     virtual ~Iterator() {}
05051
05052     virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
05053         return base_;
05054     }
05055     // Advance should not be called on beyond-of-range iterators
05056     // so no component iterators must be beyond end of range, either.
05057     virtual void Advance() {
05058         assert(!AtEnd());
05059         ++current10_;
05060         if (current10_ == endl10_) {
05061             current10_ = begin10_;
05062             ++current9_;
05063         }
05064         if (current9_ == endl9_) {
05065             current9_ = begin9_;
05066             ++current8_;
05067         }
05068         if (current8_ == endl8_) {
05069             current8_ = begin8_;
05070             ++current7_;
05071         }
05072         if (current7_ == endl7_) {
05073             current7_ = begin7_;
05074             ++current6_;
05075         }
05076         if (current6_ == endl6_) {
05077             current6_ = begin6_;
05078             ++current5_;
05079         }
05080         if (current5_ == endl5_) {
05081             current5_ = begin5_;
05082             ++current4_;
05083         }
05084         if (current4_ == endl4_) {
05085             current4_ = begin4_;
05086             ++current3_;
05087         }
05088         if (current3_ == endl3_) {
05089             current3_ = begin3_;
05090             ++current2_;
05091         }
05092         if (current2_ == endl2_) {
05093             current2_ = begin2_;
05094             ++current1_;
05095         }
05096     ComputeCurrentValue();
05097 }
05098     virtual ParamIteratorInterface<ParamType>* Clone() const {
05099         return new Iterator(*this);
05100     }
05101     virtual const ParamType* Current() const { return current_value_.get(); }
05102     virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
05103         // Having the same base generator guarantees that the other
05104         // iterator is of the same type and we can downcast.
05105         GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
05106             << "The program attempted to compare iterators "
05107             << "from different generators." << std::endl;
05108     const Iterator* typed_other =

```

```

05109     CheckedDowncastToActualType<const Iterator>(&other);
05110     // We must report iterators equal if they both point beyond their
05111     // respective ranges. That can happen in a variety of fashions,
05112     // so we have to consult AtEnd().
05113     return (AtEnd() && typed_other->AtEnd()) ||
05114     (
05115         current1_ == typed_other->current1_ &&
05116         current2_ == typed_other->current2_ &&
05117         current3_ == typed_other->current3_ &&
05118         current4_ == typed_other->current4_ &&
05119         current5_ == typed_other->current5_ &&
05120         current6_ == typed_other->current6_ &&
05121         current7_ == typed_other->current7_ &&
05122         current8_ == typed_other->current8_ &&
05123         current9_ == typed_other->current9_ &&
05124         current10_ == typed_other->current10_);
05125     }
05126
05127 private:
05128     Iterator(const Iterator& other)
05129         : base_(other.base_),
05130         begin1_(other.begin1_),
05131         end1_(other.end1_),
05132         current1_(other.current1_),
05133         begin2_(other.begin2_),
05134         end2_(other.end2_),
05135         current2_(other.current2_),
05136         begin3_(other.begin3_),
05137         end3_(other.end3_),
05138         current3_(other.current3_),
05139         begin4_(other.begin4_),
05140         end4_(other.end4_),
05141         current4_(other.current4_),
05142         begin5_(other.begin5_),
05143         end5_(other.end5_),
05144         current5_(other.current5_),
05145         begin6_(other.begin6_),
05146         end6_(other.end6_),
05147         current6_(other.current6_),
05148         begin7_(other.begin7_),
05149         end7_(other.end7_),
05150         current7_(other.current7_),
05151         begin8_(other.begin8_),
05152         end8_(other.end8_),
05153         current8_(other.current8_),
05154         begin9_(other.begin9_),
05155         end9_(other.end9_),
05156         current9_(other.current9_),
05157         begin10_(other.begin10_),
05158         end10_(other.end10_),
05159         current10_(other.current10_) {
05160     ComputeCurrentValue();
05161 }
05162
05163     void ComputeCurrentValue() {
05164         if (!AtEnd())
05165             current_value_.reset(new ParamType(*current1_, *current2_, *current3_,
05166                         *current4_, *current5_, *current6_, *current7_, *current8_,
05167                         *current9_, *current10_));
05168     }
05169     bool AtEnd() const {
05170         // We must report iterator past the end of the range when either of the
05171         // component iterators has reached the end of its range.
05172         return
05173             current1_ == end1_ ||
05174             current2_ == end2_ ||
05175             current3_ == end3_ ||
05176             current4_ == end4_ ||
05177             current5_ == end5_ ||
05178             current6_ == end6_ ||
05179             current7_ == end7_ ||
05180             current8_ == end8_ ||
05181             current9_ == end9_ ||
05182             current10_ == end10_;
05183     }
05184
05185     // No implementation - assignment is unsupported.
05186     void operator=(const Iterator& other);
05187
05188     const ParamGeneratorInterface<ParamType>*> const base_;
05189     // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
05190     // current[i]_ is the actual traversing iterator.
05191     const typename ParamGenerator<T1>::iterator begin1_;
05192     const typename ParamGenerator<T1>::iterator end1_;
05193     typename ParamGenerator<T1>::iterator current1_;
05194     const typename ParamGenerator<T2>::iterator begin2_;
05195     const typename ParamGenerator<T2>::iterator end2_;

```

```

05196     typename ParamGenerator<T2>::iterator current2_;
05197     const typename ParamGenerator<T3>::iterator begin3_;
05198     const typename ParamGenerator<T3>::iterator end3_;
05199     typename ParamGenerator<T3>::iterator current3_;
05200     const typename ParamGenerator<T4>::iterator begin4_;
05201     const typename ParamGenerator<T4>::iterator end4_;
05202     typename ParamGenerator<T4>::iterator current4_;
05203     const typename ParamGenerator<T5>::iterator begin5_;
05204     const typename ParamGenerator<T5>::iterator end5_;
05205     typename ParamGenerator<T5>::iterator current5_;
05206     const typename ParamGenerator<T6>::iterator begin6_;
05207     const typename ParamGenerator<T6>::iterator end6_;
05208     typename ParamGenerator<T6>::iterator current6_;
05209     const typename ParamGenerator<T7>::iterator begin7_;
05210     const typename ParamGenerator<T7>::iterator end7_;
05211     typename ParamGenerator<T7>::iterator current7_;
05212     const typename ParamGenerator<T8>::iterator begin8_;
05213     const typename ParamGenerator<T8>::iterator end8_;
05214     typename ParamGenerator<T8>::iterator current8_;
05215     const typename ParamGenerator<T9>::iterator begin9_;
05216     const typename ParamGenerator<T9>::iterator end9_;
05217     typename ParamGenerator<T9>::iterator current9_;
05218     const typename ParamGenerator<T10>::iterator begin10_;
05219     const typename ParamGenerator<T10>::iterator end10_;
05220     typename ParamGenerator<T10>::iterator current10_;
05221     linked_ptr<ParamType> current_value_;
05222 }; // class CartesianProductGenerator10::Iterator
05223
05224 // No implementation - assignment is unsupported.
05225 void operator=(const CartesianProductGenerator10& other);
05226
05227 const ParamGenerator<T1> g1_;
05228 const ParamGenerator<T2> g2_;
05229 const ParamGenerator<T3> g3_;
05230 const ParamGenerator<T4> g4_;
05231 const ParamGenerator<T5> g5_;
05232 const ParamGenerator<T6> g6_;
05233 const ParamGenerator<T7> g7_;
05234 const ParamGenerator<T8> g8_;
05235 const ParamGenerator<T9> g9_;
05236 const ParamGenerator<T10> g10_;
05237 }; // class CartesianProductGenerator10
05238
05239
05240 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
05241 //
05242 // Helper classes providing Combine() with polymorphic features. They allow
05243 // casting CartesianProductGeneratorN<T> to ParamGenerator<U> if T is
05244 // convertible to U.
05245 //
05246 template <class Generator1, class Generator2>
05247 class CartesianProductHolder2 {
05248 public:
05249     CartesianProductHolder2(const Generator1& g1, const Generator2& g2)
05250         : g1_(g1), g2_(g2) {}
05251     template <typename T1, typename T2>
05252     operator ParamGenerator< ::testing::tuple<T1, T2> >() const {
05253         return ParamGenerator< ::testing::tuple<T1, T2> >(
05254             new CartesianProductGenerator2<T1, T2>(
05255                 static_cast<ParamGenerator<T1>>(g1_),
05256                 static_cast<ParamGenerator<T2>>(g2_)));
05257     }
05258
05259 private:
05260     // No implementation - assignment is unsupported.
05261     void operator=(const CartesianProductHolder2& other);
05262
05263     const Generator1 g1_;
05264     const Generator2 g2_;
05265 }; // class CartesianProductHolder2
05266
05267 template <class Generator1, class Generator2, class Generator3>
05268 class CartesianProductHolder3 {
05269 public:
05270     CartesianProductHolder3(const Generator1& g1, const Generator2& g2,
05271         const Generator3& g3)
05272         : g1_(g1), g2_(g2), g3_(g3) {}
05273     template <typename T1, typename T2, typename T3>
05274     operator ParamGenerator< ::testing::tuple<T1, T2, T3> >() const {
05275         return ParamGenerator< ::testing::tuple<T1, T2, T3> >(
05276             new CartesianProductGenerator3<T1, T2, T3>(
05277                 static_cast<ParamGenerator<T1>>(g1_),
05278                 static_cast<ParamGenerator<T2>>(g2_),
05279                 static_cast<ParamGenerator<T3>>(g3_)));
05280     }
05281
05282 private:

```

```

05283 // No implementation - assignment is unsupported.
05284 void operator=(const CartesianProductHolder3& other);
05285
05286 const Generator1 g1_;
05287 const Generator2 g2_;
05288 const Generator3 g3_;
05289 }; // class CartesianProductHolder3
05290
05291 template <class Generator1, class Generator2, class Generator3,
05292     class Generator4>
05293 class CartesianProductHolder4 {
05294 public:
05295 CartesianProductHolder4(const Generator1& g1, const Generator2& g2,
05296     const Generator3& g3, const Generator4& g4)
05297     : g1_(g1), g2_(g2), g3_(g3), g4_(g4) {}
05298 template <typename T1, typename T2, typename T3, typename T4>
05299 operator ParamGenerator< ::testing::tuple<T1, T2, T3, T4> >() const {
05300     return ParamGenerator< ::testing::tuple<T1, T2, T3, T4> >(
05301         new CartesianProductGenerator4<T1, T2, T3, T4>(
05302             static_cast<ParamGenerator<T1>>(g1_),
05303             static_cast<ParamGenerator<T2>>(g2_),
05304             static_cast<ParamGenerator<T3>>(g3_),
05305             static_cast<ParamGenerator<T4>>(g4_)));
05306 }
05307
05308 private:
05309 // No implementation - assignment is unsupported.
05310 void operator=(const CartesianProductHolder4& other);
05311
05312 const Generator1 g1_;
05313 const Generator2 g2_;
05314 const Generator3 g3_;
05315 const Generator4 g4_;
05316 }; // class CartesianProductHolder4
05317
05318 template <class Generator1, class Generator2, class Generator3,
05319     class Generator4, class Generator5>
05320 class CartesianProductHolder5 {
05321 public:
05322 CartesianProductHolder5(const Generator1& g1, const Generator2& g2,
05323     const Generator3& g3, const Generator4& g4, const Generator5& g5)
05324     : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5) {}
05325 template <typename T1, typename T2, typename T3, typename T4, typename T5>
05326 operator ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5> >() const {
05327     return ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5> >(
05328         new CartesianProductGenerator5<T1, T2, T3, T4, T5>(
05329             static_cast<ParamGenerator<T1>>(g1_),
05330             static_cast<ParamGenerator<T2>>(g2_),
05331             static_cast<ParamGenerator<T3>>(g3_),
05332             static_cast<ParamGenerator<T4>>(g4_),
05333             static_cast<ParamGenerator<T5>>(g5_)));
05334 }
05335
05336 private:
05337 // No implementation - assignment is unsupported.
05338 void operator=(const CartesianProductHolder5& other);
05339
05340 const Generator1 g1_;
05341 const Generator2 g2_;
05342 const Generator3 g3_;
05343 const Generator4 g4_;
05344 const Generator5 g5_;
05345 }; // class CartesianProductHolder5
05346
05347 template <class Generator1, class Generator2, class Generator3,
05348     class Generator4, class Generator5, class Generator6>
05349 class CartesianProductHolder6 {
05350 public:
05351 CartesianProductHolder6(const Generator1& g1, const Generator2& g2,
05352     const Generator3& g3, const Generator4& g4, const Generator5& g5,
05353     const Generator6& g6)
05354     : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6) {}
05355 template <typename T1, typename T2, typename T3, typename T4, typename T5,
05356     typename T6>
05357 operator ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6> >() const {
05358     return ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6> >(
05359         new CartesianProductGenerator6<T1, T2, T3, T4, T5, T6>(
05360             static_cast<ParamGenerator<T1>>(g1_),
05361             static_cast<ParamGenerator<T2>>(g2_),
05362             static_cast<ParamGenerator<T3>>(g3_),
05363             static_cast<ParamGenerator<T4>>(g4_),
05364             static_cast<ParamGenerator<T5>>(g5_),
05365             static_cast<ParamGenerator<T6>>(g6_)));
05366 }
05367
05368 private:
05369 // No implementation - assignment is unsupported.

```

```

05370 void operator=(const CartesianProductHolder6& other);
05371
05372 const Generator1 g1_;
05373 const Generator2 g2_;
05374 const Generator3 g3_;
05375 const Generator4 g4_;
05376 const Generator5 g5_;
05377 const Generator6 g6_;
05378 }; // class CartesianProductHolder6
05379
05380 template <class Generator1, class Generator2, class Generator3,
05381     class Generator4, class Generator5, class Generator6, class Generator7>
05382 class CartesianProductHolder7 {
05383 public:
05384 CartesianProductHolder7(const Generator1& g1, const Generator2& g2,
05385     const Generator3& g3, const Generator4& g4, const Generator5& g5,
05386     const Generator6& g6, const Generator7& g7)
05387     : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7) {}
05388 template <typename T1, typename T2, typename T3, typename T4, typename T5,
05389     typename T6, typename T7>
05390 operator ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6,
05391     T7> >() const {
05392     return ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6, T7> >(
05393         new CartesianProductGenerator7<T1, T2, T3, T4, T5, T6, T7>(
05394             static_cast<ParamGenerator<T1>>(g1_),
05395             static_cast<ParamGenerator<T2>>(g2_),
05396             static_cast<ParamGenerator<T3>>(g3_),
05397             static_cast<ParamGenerator<T4>>(g4_),
05398             static_cast<ParamGenerator<T5>>(g5_),
05399             static_cast<ParamGenerator<T6>>(g6_),
05400             static_cast<ParamGenerator<T7>>(g7_)));
05401 }
05402
05403 private:
05404 // No implementation - assignment is unsupported.
05405 void operator=(const CartesianProductHolder7& other);
05406
05407 const Generator1 g1_;
05408 const Generator2 g2_;
05409 const Generator3 g3_;
05410 const Generator4 g4_;
05411 const Generator5 g5_;
05412 const Generator6 g6_;
05413 const Generator7 g7_;
05414 }; // class CartesianProductHolder7
05415
05416 template <class Generator1, class Generator2, class Generator3,
05417     class Generator4, class Generator5, class Generator6, class Generator7,
05418     class Generator8>
05419 class CartesianProductHolder8 {
05420 public:
05421 CartesianProductHolder8(const Generator1& g1, const Generator2& g2,
05422     const Generator3& g3, const Generator4& g4, const Generator5& g5,
05423     const Generator6& g6, const Generator7& g7, const Generator8& g8)
05424     : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7),
05425     g8_(g8) {}
05426 template <typename T1, typename T2, typename T3, typename T4, typename T5,
05427     typename T6, typename T7, typename T8>
05428 operator ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6, T7,
05429     T8> >() const {
05430     return ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8> >(
05431         new CartesianProductGenerator8<T1, T2, T3, T4, T5, T6, T7, T8>(
05432             static_cast<ParamGenerator<T1>>(g1_),
05433             static_cast<ParamGenerator<T2>>(g2_),
05434             static_cast<ParamGenerator<T3>>(g3_),
05435             static_cast<ParamGenerator<T4>>(g4_),
05436             static_cast<ParamGenerator<T5>>(g5_),
05437             static_cast<ParamGenerator<T6>>(g6_),
05438             static_cast<ParamGenerator<T7>>(g7_),
05439             static_cast<ParamGenerator<T8>>(g8_)));
05440 }
05441
05442 private:
05443 // No implementation - assignment is unsupported.
05444 void operator=(const CartesianProductHolder8& other);
05445
05446 const Generator1 g1_;
05447 const Generator2 g2_;
05448 const Generator3 g3_;
05449 const Generator4 g4_;
05450 const Generator5 g5_;
05451 const Generator6 g6_;
05452 const Generator7 g7_;
05453 const Generator8 g8_;
05454 }; // class CartesianProductHolder8
05455
05456 template <class Generator1, class Generator2, class Generator3,

```

```
05457     class Generator4, class Generator5, class Generator6, class Generator7,
05458     class Generator8, class Generator9>
05459 class CartesianProductHolder9 {
05460 public:
05461 CartesianProductHolder9(const Generator1& g1, const Generator2& g2,
05462     const Generator3& g3, const Generator4& g4, const Generator5& g5,
05463     const Generator6& g6, const Generator7& g7, const Generator8& g8,
05464     const Generator9& g9)
05465     : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7), g8_(g8),
05466     g9_(g9) {}
05467 template <typename T1, typename T2, typename T3, typename T4, typename T5,
05468     typename T6, typename T7, typename T8, typename T9>
05469 operator ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8,
05470     T9> >() const {
05471     return ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8,
05472         T9> >(
05473         new CartesianProductGenerator9<T1, T2, T3, T4, T5, T6, T7, T8, T9>(
05474             static_cast<ParamGenerator<T1>>(g1_),
05475             static_cast<ParamGenerator<T2>>(g2_),
05476             static_cast<ParamGenerator<T3>>(g3_),
05477             static_cast<ParamGenerator<T4>>(g4_),
05478             static_cast<ParamGenerator<T5>>(g5_),
05479             static_cast<ParamGenerator<T6>>(g6_),
05480             static_cast<ParamGenerator<T7>>(g7_),
05481             static_cast<ParamGenerator<T8>>(g8_),
05482             static_cast<ParamGenerator<T9>>(g9_)));
05483     }
05484
05485 private:
05486 // No implementation - assignment is unsupported.
05487 void operator=(const CartesianProductHolder9& other);
05488
05489 const Generator1 g1_;
05490 const Generator2 g2_;
05491 const Generator3 g3_;
05492 const Generator4 g4_;
05493 const Generator5 g5_;
05494 const Generator6 g6_;
05495 const Generator7 g7_;
05496 const Generator8 g8_;
05497 const Generator9 g9_;
05498 }; // class CartesianProductHolder9
05499
05500 template <class Generator1, class Generator2, class Generator3,
05501     class Generator4, class Generator5, class Generator6, class Generator7,
05502     class Generator8, class Generator9, class Generator10>
05503 class CartesianProductHolder10 {
05504 public:
05505 CartesianProductHolder10(const Generator1& g1, const Generator2& g2,
05506     const Generator3& g3, const Generator4& g4, const Generator5& g5,
05507     const Generator6& g6, const Generator7& g7, const Generator8& g8,
05508     const Generator9& g9, const Generator10& g10)
05509     : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7), g8_(g8),
05510     g9_(g9), g10_(g10) {}
05511 template <typename T1, typename T2, typename T3, typename T4, typename T5,
05512     typename T6, typename T7, typename T8, typename T9, typename T10>
05513 operator ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8, T9,
05514     T10> >() const {
05515     return ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8, T9,
05516         T10> >(
05517         new CartesianProductGenerator10<T1, T2, T3, T4, T5, T6, T7, T8, T9,
05518             T10>(
05519             static_cast<ParamGenerator<T1>>(g1_),
05520             static_cast<ParamGenerator<T2>>(g2_),
05521             static_cast<ParamGenerator<T3>>(g3_),
05522             static_cast<ParamGenerator<T4>>(g4_),
05523             static_cast<ParamGenerator<T5>>(g5_),
05524             static_cast<ParamGenerator<T6>>(g6_),
05525             static_cast<ParamGenerator<T7>>(g7_),
05526             static_cast<ParamGenerator<T8>>(g8_),
05527             static_cast<ParamGenerator<T9>>(g9_),
05528             static_cast<ParamGenerator<T10>>(g10_)));
05529     }
05530
05531 private:
05532 // No implementation - assignment is unsupported.
05533 void operator=(const CartesianProductHolder10& other);
05534
05535 const Generator1 g1_;
05536 const Generator2 g2_;
05537 const Generator3 g3_;
05538 const Generator4 g4_;
05539 const Generator5 g5_;
05540 const Generator6 g6_;
05541 const Generator7 g7_;
05542 const Generator8 g8_;
05543 const Generator9 g9_;
```

```
05544     const Generator10 g10_;
05545 } // class CartesianProductHolder10
05546
05547 #endif // GTEST_HAS_COMBINE
05548 } // namespace internal
05549 } // namespace testing
05550
05551
05552 #endif // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PARAM_UTIL_GENERATED_H_
```

9.43 Dokumentacja pliku

packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.h

```
#include <ctype.h>
#include <iterator>
#include <set>
#include <utility>
#include <vector>
#include "gtest/internal/gtest-internal.h"
#include "gtest/internal/gtest-linked_ptr.h"
#include "gtest/internal/gtest-port.h"
#include "gtest/gtest-printers.h"
```

Komponenty

- struct `testing::TestParamInfo< ParamType >`
- struct `testing::PrintToStringParamName`
- class `testing::internal::ParamIteratorInterface< T >`
- class `testing::internal::ParamIterator< T >`
- class `testing::internal::ParamGeneratorInterface< T >`
- class `testing::internal::ParamGenerator< T >`
- class `testing::internal::RangeGenerator< T, IncrementT >`
- class `testing::internal::ValuesInIteratorRangeGenerator< T >`
- struct `testing::internal::ParamNameGenFunc< ParamType >`
- class `testing::internal::ParameterizedTestFactory< TestClass >`
- class `testing::internal::TestMetaFactoryBase< ParamType >`
- class `testing::internal::TestMetaFactory< TestCase >`
- class `testing::internal::ParameterizedTestCaseInfoBase`
- class `testing::internal::ParameterizedTestCaseInfo< TestCase >`
- class `testing::internal::ParameterizedTestCaseRegistry`

Przestrzenie nazw

- namespace `testing`
- namespace `testing::internal`

Funkcje

- `GTEST_API_ void testing::internal::ReportInvalidTestCaseType (const char *test_case_name, CodeLocation code_location)`
- `template<class ParamType> std::string testing::internal::DefaultParamName (const TestParamInfo< ParamType > &info)`
- `template<class ParamType, class ParamNameGenFunctor> ParamNameGenFunctor testing::internal::GetParamNameGen (ParamNameGenFunctor func)`
- `template<class ParamType> ParamNameGenFunc< ParamType >::Type * testing::internal::GetParamNameGen ()`

9.44 gtest-param-util.h

[Idź do dokumentacji tego pliku.](#)

```

00001 // Copyright 2008 Google Inc.
00002 // All Rights Reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //      * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //      * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029
00030
00031 // Type and function utilities for implementing parameterized tests.
00032
00033 // GOOGLETEST_CM0001 DO NOT DELETE
00034
00035 #ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PARAM_UTIL_H_
00036 #define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PARAM_UTIL_H_
00037
00038 #include <ctype.h>
00039
00040 #include <iterator>
00041 #include <set>
00042 #include <utility>
00043 #include <vector>
00044
00045 #include "gtest/internal/gtest-internal.h"
00046 #include "gtest/internal/gtest-linked_ptr.h"
00047 #include "gtest/internal/gtest-port.h"
00048 #include "gtest/gtest-printers.h"
00049
00050 namespace testing {
00051
00052 // Input to a parameterized test name generator, describing a test parameter.
00053 // Consists of the parameter value and the integer parameter index.
00054 template <class ParamType>
00055 struct TestParamInfo {
00056     TestParamInfo(const ParamType& a_param, size_t an_index) :
00057         param(a_param),
00058         index(an_index) {}
00059     ParamType param;
00060     size_t index;
00061 };
00062

```

```

00063 // A builtin parameterized test name generator which returns the result of
00064 // testing::PrintToString.
00065 struct PrintToStringParamName {
00066     template <class ParamType>
00067     std::string operator()(const TestParamInfo<ParamType>& info) const {
00068         return PrintToString(info.param);
00069     }
00070 };
00071
00072 namespace internal {
00073
00074 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
00075 //
00076 // Outputs a message explaining invalid registration of different
00077 // fixture class for the same test case. This may happen when
00078 // TEST_P macro is used to define two tests with the same name
00079 // but in different namespaces.
00080 GTEST_API_ void ReportInvalidTestCaseType(const char* test_case_name,
00081                                              CodeLocation code_location);
00082
00083 template <typename> class ParamGeneratorInterface;
00084 template <typename> class ParamGenerator;
00085
00086 // Interface for iterating over elements provided by an implementation
00087 // of ParamGeneratorInterface<T>.
00088 template <typename T>
00089 class ParamIteratorInterface {
00090 public:
00091     virtual ~ParamIteratorInterface() {}
00092     // A pointer to the base generator instance.
00093     // Used only for the purposes of iterator comparison
00094     // to make sure that two iterators belong to the same generator.
00095     virtual const ParamGeneratorInterface<T>* BaseGenerator() const = 0;
00096     // Advances iterator to point to the next element
00097     // provided by the generator. The caller is responsible
00098     // for not calling Advance() on an iterator equal to
00099     // BaseGenerator()->End().
00100    virtual void Advance() = 0;
00101    // Clones the iterator object. Used for implementing copy semantics
00102    // of ParamIterator<T>.
00103    virtual ParamIteratorInterface* Clone() const = 0;
00104    // Dereferences the current iterator and provides (read-only) access
00105    // to the pointed value. It is the caller's responsibility not to call
00106    // Current() on an iterator equal to BaseGenerator()->End().
00107    // Used for implementing ParamGenerator<T>::operator*().
00108    virtual const T* Current() const = 0;
00109    // Determines whether the given iterator and other point to the same
00110    // element in the sequence generated by the generator.
00111    // Used for implementing ParamGenerator<T>::operator==().
00112    virtual bool Equals(const ParamIteratorInterface& other) const = 0;
00113 };
00114
00115 // Class iterating over elements provided by an implementation of
00116 // ParamGeneratorInterface<T>. It wraps ParamIteratorInterface<T>
00117 // and implements the const forward iterator concept.
00118 template <typename T>
00119 class ParamIterator {
00120 public:
00121     typedef T value_type;
00122     typedef const T& reference;
00123     typedef ptrdiff_t difference_type;
00124
00125     // ParamIterator assumes ownership of the impl_ pointer.
00126     ParamIterator(const ParamIterator& other) : impl_(other.impl_->Clone()) {}
00127     ParamIterator& operator=(const ParamIterator& other) {
00128         if (this != &other)
00129             impl_.reset(other.impl_->Clone());
00130         return *this;
00131     }
00132
00133     const T& operator*() const { return *impl_->Current(); }
00134     const T* operator->() const { return impl_->Current(); }
00135     // Prefix version of operator++.
00136     ParamIterator& operator++() {
00137         impl_->Advance();
00138         return *this;
00139     }
00140     // Postfix version of operator++.
00141     ParamIterator operator++(int /*unused*/) {
00142         ParamIteratorInterface<T>* clone = impl_->Clone();
00143         impl_->Advance();
00144         return ParamIterator(clone);
00145     }
00146     bool operator==(const ParamIterator& other) const {
00147         return impl_.get() == other.impl_.get() || impl_->Equals(*other.impl_);
00148     }
00149     bool operator!=(const ParamIterator& other) const {

```

```

00150     return !(*this == other);
00151 }
00152
00153 private:
00154     friend class ParamGenerator<T>;
00155     explicit ParamIterator(ParamIteratorInterface<T>* impl) : impl_(impl) {}
00156     scoped_ptr<ParamIteratorInterface<T> > impl_;
00157 };
00158
00159 // ParamGeneratorInterface<T> is the binary interface to access generators
00160 // defined in other translation units.
00161 template <typename T>
00162 class ParamGeneratorInterface {
00163 public:
00164     typedef T ParamType;
00165
00166     virtual ~ParamGeneratorInterface() {}
00167
00168     // Generator interface definition
00169     virtual ParamIteratorInterface<T>* Begin() const = 0;
00170     virtual ParamIteratorInterface<T>* End() const = 0;
00171 };
00172
00173 // Wraps ParamGeneratorInterface<T> and provides general generator syntax
00174 // compatible with the STL Container concept.
00175 // This class implements copy initialization semantics and the contained
00176 // ParamGeneratorInterface<T> instance is shared among all copies
00177 // of the original object. This is possible because that instance is immutable.
00178 template<typename T>
00179 class ParamGenerator {
00180 public:
00181     typedef ParamIterator<T> iterator;
00182
00183     explicit ParamGenerator(ParamGeneratorInterface<T>* impl) : impl_(impl) {}
00184     ParamGenerator(const ParamGenerator& other) : impl_(other.impl_) {}
00185
00186     ParamGenerator& operator=(const ParamGenerator& other) {
00187         impl_ = other.impl_;
00188         return *this;
00189     }
00190
00191     iterator begin() const { return iterator(impl_->Begin()); }
00192     iterator end() const { return iterator(impl_->End()); }
00193
00194 private:
00195     linked_ptr<const ParamGeneratorInterface<T> > impl_;
00196 };
00197
00198 // Generates values from a range of two comparable values. Can be used to
00199 // generate sequences of user-defined types that implement operator+() and
00200 // operator<().
00201 // This class is used in the Range() function.
00202 template <typename T, typename IncrementT>
00203 class RangeGenerator : public ParamGeneratorInterface<T> {
00204 public:
00205     RangeGenerator(T begin, T end, IncrementT step)
00206         : begin_(begin), end_(end),
00207           step_(step), end_index_(CalculateEndIndex(begin, end, step)) {}
00208     virtual ~RangeGenerator() {}
00209
00210     virtual ParamIteratorInterface<T>* Begin() const {
00211         return new Iterator(this, begin_, 0, step_);
00212     }
00213     virtual ParamIteratorInterface<T>* End() const {
00214         return new Iterator(this, end_, end_index_, step_);
00215     }
00216
00217 private:
00218     class Iterator : public ParamIteratorInterface<T> {
00219     public:
00220         Iterator(const ParamGeneratorInterface<T>* base, T value, int index,
00221                  IncrementT step)
00222             : base_(base), value_(value), index_(index), step_(step) {}
00223         virtual ~Iterator() {}
00224
00225         virtual const ParamGeneratorInterface<T>* BaseGenerator() const {
00226             return base_;
00227         }
00228         virtual void Advance() {
00229             value_ = static_cast<T>(value_ + step_);
00230             index_++;
00231         }
00232         virtual ParamIteratorInterface<T>* Clone() const {
00233             return new Iterator(*this);
00234         }
00235         virtual const T* Current() const { return &value_; }
00236         virtual bool Equals(const ParamIteratorInterface<T>& other) const {

```

```

00237 // Having the same base generator guarantees that the other
00238 // iterator is of the same type and we can downcast.
00239 GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
00240   « "The program attempted to compare iterators "
00241   « "from different generators." « std::endl;
00242 const int other_index =
00243   CheckedDowncastToActualType<const Iterator>(&other)->index_;
00244 return index_ == other_index;
00245 }
00246
00247 private:
00248   Iterator(const Iterator& other)
00249     : ParamIteratorInterface<T>(),
00250       base_(other.base_), value_(other.value_), index_(other.index_),
00251       step_(other.step_) {}
00252
00253 // No implementation - assignment is unsupported.
00254 void operator=(const Iterator& other);
00255
00256 const ParamGeneratorInterface<T>* const base_;
00257 T value_;
00258 int index_;
00259 const IncrementT step_;
00260 }; // class RangeGenerator::Iterator
00261
00262 static int CalculateEndIndex(const T& begin,
00263                               const T& end,
00264                               const IncrementT& step) {
00265   int end_index = 0;
00266   for (T i = begin; i < end; i = static_cast<T>(i + step))
00267     end_index++;
00268   return end_index;
00269 }
00270
00271 // No implementation - assignment is unsupported.
00272 void operator=(const RangeGenerator& other);
00273
00274 const T begin_;
00275 const T end_;
00276 const IncrementT step_;
00277 // The index for the end() iterator. All the elements in the generated
00278 // sequence are indexed (0-based) to aid iterator comparison.
00279 const int end_index_;
00280 }; // class RangeGenerator
00281
00282
00283 // Generates values from a pair of STL-style iterators. Used in the
00284 // ValuesIn() function. The elements are copied from the source range
00285 // since the source can be located on the stack, and the generator
00286 // is likely to persist beyond that stack frame.
00287 template <typename T>
00288 class ValuesInIteratorRangeGenerator : public ParamGeneratorInterface<T> {
00289 public:
00290   template <typename ForwardIterator>
00291   ValuesInIteratorRangeGenerator(ForwardIterator begin, ForwardIterator end)
00292     : container_(begin, end) {}
00293   virtual ~ValuesInIteratorRangeGenerator() {}
00294
00295   virtual ParamIteratorInterface<T>* Begin() const {
00296     return new Iterator(this, container_.begin());
00297   }
00298   virtual ParamIteratorInterface<T>* End() const {
00299     return new Iterator(this, container_.end());
00300   }
00301
00302 private:
00303   typedef typename ::std::vector<T> ContainerType;
00304
00305   class Iterator : public ParamIteratorInterface<T> {
00306   public:
00307     Iterator(const ParamGeneratorInterface<T>* base,
00308              typename ContainerType::const_iterator iterator)
00309       : base_(base), iterator_(iterator) {}
00310     virtual ~Iterator() {}
00311
00312     virtual const ParamGeneratorInterface<T>* BaseGenerator() const {
00313       return base_;
00314     }
00315     virtual void Advance() {
00316       ++iterator_;
00317       value_.reset();
00318     }
00319     virtual ParamIteratorInterface<T>* Clone() const {
00320       return new Iterator(*this);
00321     }
00322   // We need to use cached value referenced by iterator_
00323   // because *iterator_
00324   // can return a temporary object (and of type other then T), so just

```

```

00324     // having "return &iterator_;" doesn't work.
00325     // value_ is updated here and not in Advance() because Advance()
00326     // can advance iterator_ beyond the end of the range, and we cannot
00327     // detect that fact. The client code, on the other hand, is
00328     // responsible for not calling Current() on an out-of-range iterator.
00329     virtual const T* Current() const {
00330         if (value_.get() == NULL)
00331             value_.reset(new T(*iterator_));
00332         return value_.get();
00333     }
00334     virtual bool Equals(const ParamIteratorInterface<T>& other) const {
00335         // Having the same base generator guarantees that the other
00336         // iterator is of the same type and we can downcast.
00337         GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
00338         « "The program attempted to compare iterators "
00339         « "from different generators." « std::endl;
00340         return iterator_ ==
00341             CheckedDowncastToActualType<const Iterator>(&other)->iterator_;
00342     }
00343
00344 private:
00345     Iterator(const Iterator& other)
00346         // The explicit constructor call suppresses a false warning
00347         // emitted by gcc when supplied with the -Wextra option.
00348         : ParamIteratorInterface<T>(),
00349             base_(other.base_),
00350             iterator_(other.iterator_) {}
00351
00352     const ParamGeneratorInterface<T>* const base_;
00353     typename ContainerType::const_iterator iterator_;
00354     // A cached value of *iterator_. We keep it here to allow access by
00355     // pointer in the wrapping iterator's operator->().
00356     // value_ needs to be mutable to be accessed in Current().
00357     // Use of scoped_ptr helps manage cached value's lifetime,
00358     // which is bound by the lifespan of the iterator itself.
00359     mutable scoped_ptr<const T> value_;
00360 }; // class ValuesInIteratorRangeGenerator::Iterator
00361
00362 // No implementation - assignment is unsupported.
00363 void operator=(const ValuesInIteratorRangeGenerator& other);
00364
00365 const ContainerType container_;
00366 }; // class ValuesInIteratorRangeGenerator
00367
00368 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
00369 //
00370 // Default parameterized test name generator, returns a string containing the
00371 // integer test parameter index.
00372 template <class ParamType>
00373 std::string DefaultParamName(const TestParamInfo<ParamType>& info) {
00374     Message name_stream;
00375     name_stream < info.index;
00376     return name_stream.GetString();
00377 }
00378
00379 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
00380 //
00381 // Parameterized test name overload helpers, which help the
00382 // INSTANTIATE_TEST_CASE_P macro choose between the default parameterized
00383 // test name generator and user param name generator.
00384 template <class ParamType, class ParamNameGenFunctor>
00385 ParamNameGenFunctor GetParamNameGen(ParamNameGenFunctor func) {
00386     return func;
00387 }
00388
00389 template <class ParamType>
00390 struct ParamNameGenFunc {
00391     typedef std::string Type(const TestParamInfo<ParamType>&);
00392 };
00393
00394 template <class ParamType>
00395 typename ParamNameGenFunc<ParamType>::Type *GetParamNameGen() {
00396     return DefaultParamName;
00397 }
00398
00399 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
00400 //
00401 // Stores a parameter value and later creates tests parameterized with that
00402 // value.
00403 template <class TestClass>
00404 class ParameterizedTestFactory : public TestFactoryBase {
00405 public:
00406     typedef typename TestClass::ParamType ParamType;
00407     explicit ParameterizedTestFactory(ParamType parameter) :
00408         parameter_(parameter) {}
00409     virtual Test* CreateTest() {
00410         TestClass::SetParam(&parameter_);

```

```
00411     return new TestClass();
00412 }
00413
00414 private:
00415 const ParamType parameter_;
00416
00417 GTEST_DISALLOW_COPY_AND_ASSIGN_(ParameterizedTestFactory);
00418 };
00419
00420 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
00421 //
00422 // TestMetaFactoryBase is a base class for meta-factories that create
00423 // test factories for passing into MakeAndRegisterTestInfo function.
00424 template <class ParamType>
00425 class TestMetaFactoryBase {
00426 public:
00427     virtual ~TestMetaFactoryBase() {}
00428
00429     virtual TestFactoryBase* CreateTestFactory(ParamType parameter) = 0;
00430 };
00431
00432 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
00433 //
00434 // TestMetaFactory creates test factories for passing into
00435 // MakeAndRegisterTestInfo function. Since MakeAndRegisterTestInfo receives
00436 // ownership of test factory pointer, same factory object cannot be passed
00437 // into that method twice. But ParameterizedTestCaseInfo is going to call
00438 // it for each Test/Parameter value combination. Thus it needs meta factory
00439 // creator class.
00440 template <class TestCase>
00441 class TestMetaFactory
00442     : public TestMetaFactoryBase<typename TestCase::ParamType> {
00443 public:
00444     typedef typename TestCase::ParamType ParamType;
00445
00446     TestMetaFactory() {}
00447
00448     virtual TestFactoryBase* CreateTestFactory(ParamType parameter) {
00449         return new ParameterizedTestFactory<TestCase>(parameter);
00450     }
00451
00452 private:
00453     GTEST_DISALLOW_COPY_AND_ASSIGN_(TestMetaFactory);
00454 };
00455
00456 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
00457 //
00458 // ParameterizedTestCaseInfoBase is a generic interface
00459 // to ParameterizedTestCaseInfo classes. ParameterizedTestCaseInfoBase
00460 // accumulates test information provided by TEST_P macro invocations
00461 // and generators provided by INSTANTIATE_TEST_CASE_P macro invocations
00462 // and uses that information to register all resulting test instances
00463 // in RegisterTests method. The ParameterizeTestCaseRegistry class holds
00464 // a collection of pointers to the ParameterizedTestCaseInfo objects
00465 // and calls RegisterTests() on each of them when asked.
00466 class ParameterizedTestCaseInfoBase {
00467 public:
00468     virtual ~ParameterizedTestCaseInfoBase() {}
00469
00470     // Base part of test case name for display purposes.
00471     virtual const std::string& GetTestCaseName() const = 0;
00472     // Test case id to verify identity.
00473     virtual TypeId GetTestCaseTypeId() const = 0;
00474     // UnitTest class invokes this method to register tests in this
00475     // test case right before running them in RUN_ALL_TESTS macro.
00476     // This method should not be called more than once on any single
00477     // instance of a ParameterizedTestCaseInfoBase derived class.
00478     virtual void RegisterTests() = 0;
00479
00480 protected:
00481     ParameterizedTestCaseInfoBase() {}
00482
00483 private:
00484     GTEST_DISALLOW_COPY_AND_ASSIGN_(ParameterizedTestCaseInfoBase);
00485 };
00486
00487 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
00488 //
00489 // ParameterizedTestCaseInfo accumulates tests obtained from TEST_P
00490 // macro invocations for a particular test case and generators
00491 // obtained from INSTANTIATE_TEST_CASE_P macro invocations for that
00492 // test case. It registers tests with all values generated by all
00493 // generators when asked.
00494 template <class TestCase>
00495 class ParameterizedTestCaseInfo : public ParameterizedTestCaseInfoBase {
00496 public:
00497     // ParamType and GeneratorCreationFunc are private types but are required
```

```

00498 // for declarations of public methods AddTestPattern() and
00499 // AddTestCaseInstantiation().
00500 typedef typename TestCase::ParamType ParamType;
00501 // A function that returns an instance of appropriate generator type.
00502 typedef ParamGenerator<ParamType>(GeneratorCreationFunc) ();
00503 typedef typename ParamNameGenFunc<ParamType>::Type ParamNameGeneratorFunc;
00504
00505 explicit ParameterizedTestCaseInfo(
00506     const char* name, CodeLocation code_location)
00507     : test_case_name_(name), code_location_(code_location) {}
00508
00509 // Test case base name for display purposes.
00510 virtual const std::string& GetTestCaseName() const { return test_case_name_; }
00511 // Test case id to verify identity.
00512 virtual TypeId GetTestCaseTypeId() const { return GetTypeId<TestCase>(); }
00513 // TEST_P macro uses AddTestPattern() to record information
00514 // about a single test in a LocalTestInfo structure.
00515 // test_case_name is the base name of the test case (without invocation
00516 // prefix). test_base_name is the name of an individual test without
00517 // parameter index. For the test SequenceA/FooTest.DoBar/1 FooTest is
00518 // test case base name and DoBar is test base name.
00519 void AddTestPattern(const char* test_case_name,
00520                     const char* test_base_name,
00521                     TestMetaFactoryBase<ParamType>* meta_factory) {
00522     tests_.push_back(linked_ptr<TestInfo>(new TestInfo(test_case_name,
00523                                                 test_base_name,
00524                                                 meta_factory)));
00525 }
00526 // INSTANTIATE_TEST_CASE_P macro uses AddGenerator() to record information
00527 // about a generator.
00528 int AddTestCaseInstantiation(const std::string& instantiation_name,
00529                             GeneratorCreationFunc* func,
00530                             ParamNameGeneratorFunc* name_func,
00531                             const char* file, int line) {
00532     instantiations_.push_back(
00533         InstantiationInfo(instantiation_name, func, name_func, file, line));
00534     return 0; // Return value used only to run this method in namespace scope.
00535 }
00536 // UnitTest class invokes this method to register tests in this test case
00537 // test cases right before running tests in RUN_ALL_TESTS macro.
00538 // This method should not be called more then once on any single
00539 // instance of a ParameterizedTestCaseInfoBase derived class.
00540 // UnitTest has a guard to prevent from calling this method more then once.
00541 virtual void RegisterTests() {
00542     for (typename TestInfoContainer::iterator test_it = tests_.begin();
00543           test_it != tests_.end(); ++test_it) {
00544         linked_ptr<TestInfo> test_info = *test_it;
00545         for (typename InstantiationContainer::iterator gen_it =
00546               instantiations_.begin(); gen_it != instantiations_.end();
00547               ++gen_it) {
00548             const std::string& instantiation_name = gen_it->name;
00549             ParamGenerator<ParamType> generator((*gen_it->generator)());
00550             ParamNameGeneratorFunc* name_func = gen_it->name_func;
00551             const char* file = gen_it->file;
00552             int line = gen_it->line;
00553
00554             std::string test_case_name;
00555             if (!instantiation_name.empty())
00556                 test_case_name = instantiation_name + "/";
00557             test_case_name += test_info->test_case_base_name;
00558
00559             size_t i = 0;
00560             std::set<std::string> test_param_names;
00561             for (typename ParamGenerator<ParamType>::iterator param_it =
00562                   generator.begin();
00563                   param_it != generator.end(); ++param_it, ++i) {
00564                 Message test_name_stream;
00565
00566                 std::string param_name = name_func(
00567                     TestParamInfo<ParamType>(*param_it, i));
00568
00569                 GTEST_CHECK_(IsValidParamName(param_name))
00570                     << "Parameterized test name '" << param_name
00571                     << "' is invalid, in " << file
00572                     << " line " << line << std::endl;
00573
00574                 GTEST_CHECK_(test_param_names.count(param_name) == 0)
00575                     << "Duplicate parameterized test name '" << param_name
00576                     << "', in " << file << " line " << line << std::endl;
00577
00578                 test_param_names.insert(param_name);
00579
00580                 test_name_stream << test_info->test_base_name << "/" << param_name;
00581                 MakeAndRegisterTestInfo(
00582                     test_case_name.c_str(),
00583                     test_name_stream.GetString().c_str(),
00584                     NULL, // No type parameter.

```

```

00585         PrintToString(*param_it).c_str(),
00586         code_location_,
00587         GetTestCaseTypeId(),
00588         TestCase::SetUpTestCase,
00589         TestCase::TearDownTestCase,
00590         test_info->test_meta_factory->CreateTestFactory(*param_it));
00591     } // for param_it
00592   } // for gen_it
00593 } // for test_it
00594 } // RegisterTests
00595
00596 private:
00597 // LocalTestInfo structure keeps information about a single test registered
00598 // with TEST_P macro.
00599 struct TestInfo {
00600     TestInfo(const char* a_test_case_base_name,
00601             const char* a_test_base_name,
00602             TestMetaFactoryBase<ParamType>* a_test_meta_factory) :
00603         test_case_base_name(a_test_case_base_name),
00604         test_base_name(a_test_base_name),
00605         test_meta_factory(a_test_meta_factory) {}
00606
00607     const std::string test_case_base_name;
00608     const std::string test_base_name;
00609     const scoped_ptr<TestMetaFactoryBase<ParamType> > test_meta_factory;
00610 };
00611 typedef ::std::vector<linked_ptr<TestInfo> > TestInfoContainer;
00612 // Records data received from INSTANTIATE_TEST_CASE_P macros:
00613 // <Instantiation name, Sequence generator creation function,
00614 // Name generator function, Source file, Source line>
00615 struct InstantiationInfo {
00616     InstantiationInfo(const std::string &name_in,
00617                         GeneratorCreationFunc* generator_in,
00618                         ParamNameGeneratorFunc* name_func_in,
00619                         const char* file_in,
00620                         int line_in)
00621         : name(name_in),
00622           generator(generator_in),
00623           name_func(name_func_in),
00624           file(file_in),
00625           line(line_in) {}
00626
00627     std::string name;
00628     GeneratorCreationFunc* generator;
00629     ParamNameGeneratorFunc* name_func;
00630     const char* file;
00631     int line;
00632 };
00633 typedef ::std::vector<InstantiationInfo> InstantiationContainer;
00634
00635 static bool IsValidParamName(const std::string& name) {
00636     // Check for empty string
00637     if (name.empty())
00638         return false;
00639
00640     // Check for invalid characters
00641     for (std::string::size_type index = 0; index < name.size(); ++index) {
00642         if (!isalnum(name[index]) && name[index] != '_')
00643             return false;
00644     }
00645
00646     return true;
00647 }
00648
00649 const std::string test_case_name_;
00650 CodeLocation code_location_;
00651 TestInfoContainer tests_;
00652 InstantiationContainer instantiations_;
00653
00654 GTEST_DISALLOW_COPY_AND_ASSIGN_(ParameterizedTestCaseInfo);
00655 }; // class ParameterizedTestCaseInfo
00656
00657 // INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
00658 //
00659 // ParameterizedTestCaseRegistry contains a map of ParameterizedTestCaseInfoBase
00660 // classes accessed by test case names. TEST_P and INSTANTIATE_TEST_CASE_P
00661 // macros use it to locate their corresponding ParameterizedTestCaseInfo
00662 // descriptors.
00663 class ParameterizedTestCaseRegistry {
00664 public:
00665     ParameterizedTestCaseRegistry() {}
00666     ~ParameterizedTestCaseRegistry() {
00667         for (TestCaseInfoContainer::iterator it = test_case_infos_.begin();
00668              it != test_case_infos_.end(); ++it) {
00669             delete *it;
00670         }
00671     }

```

```
00672
00673 // Looks up or creates and returns a structure containing information about
00674 // tests and instantiations of a particular test case.
00675 template <class TestCase>
00676 ParameterizedTestCaseInfo<TestCase>* GetTestCasePatternHolder(
00677     const char* test_case_name,
00678     CodeLocation code_location) {
00679     ParameterizedTestCaseInfo<TestCase>* typed_test_info = NULL;
00680     for (TestCaseInfoContainer::iterator it = test_case_infos_.begin();
00681         it != test_case_infos_.end(); ++it) {
00682         if ((*it)->GetTestCaseName() == test_case_name) {
00683             if ((*it)->GetTestCaseTypeId() != GetTypeId<TestCase>()) {
00684                 // Complain about incorrect usage of Google Test facilities
00685                 // and terminate the program since we cannot guaranty correct
00686                 // test case setup and tear-down in this case.
00687                 ReportInvalidTestCaseType(test_case_name, code_location);
00688                 posix::Abort();
00689             } else {
00690                 // At this point we are sure that the object we found is of the same
00691                 // type we are looking for, so we downcast it to that type
00692                 // without further checks.
00693                 typed_test_info = CheckedDowncastToActualType<
00694                     ParameterizedTestCaseInfo<TestCase>>(*it);
00695             }
00696             break;
00697         }
00698     }
00699     if (typed_test_info == NULL) {
00700         typed_test_info = new ParameterizedTestCaseInfo<TestCase>(
00701             test_case_name, code_location);
00702         test_case_infos_.push_back(typed_test_info);
00703     }
00704     return typed_test_info;
00705 }
00706 void RegisterTests() {
00707     for (TestCaseInfoContainer::iterator it = test_case_infos_.begin();
00708         it != test_case_infos_.end(); ++it) {
00709         (*it)->RegisterTests();
00710     }
00711 }
00712
00713 private:
00714     typedef ::std::vector<ParameterizedTestCaseInfoBase*> TestCaseInfoContainer;
00715
00716     TestCaseInfoContainer test_case_infos_;
00717
00718     GTEST_DISALLOW_COPY_AND_ASSIGN_(ParameterizedTestCaseRegistry);
00719 };
00720
00721 } // namespace internal
00722 } // namespace testing
00723
00724 #endif // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PARAM_UTIL_H_
```

9.45 Dokumentacja pliku packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-port-arch.h

9.46 gtest-port-arch.h

[Idź do dokumentacji tego pliku.](#)

```
00001 // Copyright 2015, Google Inc.
00002 // All rights reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //     * Redistributions of source code must retain the above copyright
00009 //       notice, this list of conditions and the following disclaimer.
00010 //     * Redistributions in binary form must reproduce the above
00011 //       copyright notice, this list of conditions and the following disclaimer
00012 //       in the documentation and/or other materials provided with the
00013 //       distribution.
00014 //     * Neither the name of Google Inc. nor the names of its
```

```
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 // The Google C++ Testing and Mocking Framework (Google Test)
00031 //
00032 // This header file defines the GTEST_OS_* macro.
00033 // It is separate from gtest-port.h so that custom/gtest-port.h can include it.
00034
00035 #ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PORT_ARCH_H_
00036 #define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PORT_ARCH_H_
00037
00038 // Determines the platform on which Google Test is compiled.
00039 #ifdef __CYGWIN__
00040 # define GTEST_OS_CYGWIN 1
00041 #elif defined __SYMBIAN32__
00042 # define GTEST_OS_SYMBIAN 1
00043 #elif defined _WIN32
00044 # define GTEST_OS_WINDOWS 1
00045 # ifdef _WIN32_WCE
00046 # define GTEST_OS_WINDOWS_MOBILE 1
00047 # elif defined(_MINGW_) || defined(_MINGW32_)
00048 # define GTEST_OS_WINDOWS_MINGW 1
00049 # elif defined(WINAPI_FAMILY)
00050 # include <winapifamily.h>
00051 # if WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_DESKTOP)
00052 # define GTEST_OS_WINDOWS_DESKTOP 1
00053 # elif WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_PHONE_APP)
00054 # define GTEST_OS_WINDOWS_PHONE 1
00055 # elif WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_APP)
00056 # define GTEST_OS_WINDOWS_RT 1
00057 # elif WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_TV_TITLE)
00058 # define GTEST_OS_WINDOWS_PHONE 1
00059 # define GTEST_OS_WINDOWS_TV_TITLE 1
00060 # else
00061     // WINAPI_FAMILY defined but no known partition matched.
00062     // Default to desktop.
00063 # define GTEST_OS_WINDOWS_DESKTOP 1
00064 # endif
00065 # else
00066 # define GTEST_OS_WINDOWS_DESKTOP 1
00067 # endif // _WIN32_WCE
00068 #elif defined __APPLE__
00069 # define GTEST_OS_MAC 1
00070 # if TARGET_OS_IPHONE
00071 # define GTEST_OS_IOS 1
00072 # endif
00073 #elif defined __FreeBSD__
00074 # define GTEST_OS_FREEBSD 1
00075 #elif defined __Fuchsia__
00076 # define GTEST_OS_FUCHSIA 1
00077 #elif defined __linux__
00078 # define GTEST_OS_LINUX 1
00079 # if defined __ANDROID__
00080 # define GTEST_OS_LINUX_ANDROID 1
00081 # endif
00082 #elif defined __MVS__
00083 # define GTEST_OS_ZOS 1
00084 #elif defined(__sun) && defined(__SVR4)
00085 # define GTEST_OS_SOLARIS 1
00086 #elif defined(__AIX)
00087 # define GTEST_OS_AIX 1
00088 #elif defined(__hpx)
00089 # define GTEST_OS_HPUX 1
00090 #elif defined __native_client__
00091 # define GTEST_OS_NACL 1
00092 #elif defined __NetBSD__
00093 # define GTEST_OS_NETBSD 1
00094 #elif defined __OpenBSD__
00095 # define GTEST_OS_OPENBSD 1
00096 #elif defined __QNX__
00097 # define GTEST_OS_QNX 1
00098 #endif // __CYGWIN__
00099
00100#endif // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PORT_ARCH_H_
```

9.47 Dokumentacja pliku packages/Microsoft.goolgelet.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-string.h

```
#include <string.h>
#include <string>
#include "gtest/internal/gtest-port.h"
```

Komponenty

- class testing::internal::String

Przestrzenie nazw

- namespace testing
- namespace testing::internal

Funkcje

- GTEST_API_ std::string testing::internal::StringStreamToString (::std::stringstream *stream)

9.48 gtest-string.h

[Idź do dokumentacji tego pliku.](#)

```
00001 // Copyright 2005, Google Inc.
00002 // All rights reserved.
00003 //
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //      * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //      * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 // The Google C++ Testing and Mocking Framework (Google Test)
00031 //
00032 // This header file declares the String class and functions used internally by
00033 // Google Test. They are subject to change without notice. They should not be used
00034 // by code external to Google Test.
00035 //
00036 // This header file is #included by gtest-internal.h.
00037 // It should not be #included by other files.
```

```
00038
00039 // GOOGLETEST_CM0001 DO NOT DELETE
00040
00041 #ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_STRING_H_
00042 #define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_STRING_H_
00043
00044 #ifdef __BORLANDC__
00045 // string.h is not guaranteed to provide strcpy on C++ Builder.
00046 # include <mem.h>
00047 #endif
00048
00049 #include <string.h>
00050 #include <string>
00051
00052 #include "gtest/internal/gtest-port.h"
00053
00054 namespace testing {
00055 namespace internal {
00056
00057 // String - an abstract class holding static string utilities.
00058 class GTEST_API_ String {
00059 public:
00060 // Static utility methods
00061
00062 // Clones a 0-terminated C string, allocating memory using new. The
00063 // caller is responsible for deleting the return value using
00064 // delete[]. Returns the cloned string, or NULL if the input is
00065 // NULL.
00066 //
00067 // This is different from strdup() in string.h, which allocates
00068 // memory using malloc().
00069 static const char* CloneCString(const char* c_str);
00070
00071 #if GTEST_OS_WINDOWS_MOBILE
00072 // Windows CE does not have the 'ANSI' versions of Win32 APIs. To be
00073 // able to pass strings to Win32 APIs on CE we need to convert them
00074 // to 'Unicode', UTF-16.
00075
00076 // Creates a UTF-16 wide string from the given ANSI string, allocating
00077 // memory using new. The caller is responsible for deleting the return
00078 // value using delete[]. Returns the wide string, or NULL if the
00079 // input is NULL.
00080 //
00081 // The wide string is created using the ANSI codepage (CP_ACP) to
00082 // match the behaviour of the ANSI versions of Win32 calls and the
00083 // C runtime.
00084 static LPCWSTR AnsiToUtf16(const char* c_str);
00085
00086 // Creates an ANSI string from the given wide string, allocating
00087 // memory using new. The caller is responsible for deleting the return
00088 // value using delete[]. Returns the ANSI string, or NULL if the
00089 // input is NULL.
00090 //
00091 // The returned string is created using the ANSI codepage (CP_ACP) to
00092 // match the behaviour of the ANSI versions of Win32 calls and the
00093 // C runtime.
00094 static const char* Utf16ToAnsi(LPCWSTR utf16_str);
00095 #endif
00096
00097 // Compares two C strings. Returns true iff they have the same content.
00098 //
00099 // Unlike strcmp(), this function can handle NULL argument(s). A
00100 // NULL C string is considered different to any non-NULL C string,
00101 // including the empty string.
00102 static bool CStringEquals(const char* lhs, const char* rhs);
00103
00104 // Converts a wide C string to a String using the UTF-8 encoding.
00105 // NULL will be converted to "(null)". If an error occurred during
00106 // the conversion, "(failed to convert from wide string)" is
00107 // returned.
00108 static std::string ShowWideCString(const wchar_t* wide_c_str);
00109
00110 // Compares two wide C strings. Returns true iff they have the same
00111 // content.
00112 //
00113 // Unlike wcscmp(), this function can handle NULL argument(s). A
00114 // NULL C string is considered different to any non-NULL C string,
00115 // including the empty string.
00116 static bool WideCStringEquals(const wchar_t* lhs, const wchar_t* rhs);
00117
00118 // Compares two C strings, ignoring case. Returns true iff they
00119 // have the same content.
00120 //
00121 // Unlike strcasecmp(), this function can handle NULL argument(s).
00122 // A NULL C string is considered different to any non-NULL C string,
00123 // including the empty string.
00124 static bool CaseInsensitiveCStringEquals(const char* lhs,
```

```
00125                                     const char* rhs);  
00126  
00127 // Compares two wide C strings, ignoring case. Returns true iff they  
00128 // have the same content.  
00129 //  
00130 // Unlike wcscasecmp(), this function can handle NULL argument(s).  
00131 // A NULL C string is considered different to any non-NULL wide C string,  
00132 // including the empty string.  
00133 // NB: The implementations on different platforms slightly differ.  
00134 // On windows, this method uses _wcsicmp which compares according to LC_CTYPE  
00135 // environment variable. On GNU platform this method uses wcscasecmp  
00136 // which compares according to LC_CTYPE category of the current locale.  
00137 // On MacOS X, it uses towlower, which also uses LC_CTYPE category of the  
00138 // current locale.  
00139 static bool CaseInsensitiveWideCStringEquals(const wchar_t* lhs,  
00140                                              const wchar_t* rhs);  
00141  
00142 // Returns true iff the given string ends with the given suffix, ignoring  
00143 // case. Any string is considered to end with an empty suffix.  
00144 static bool EndsWithCaseInsensitive(  
00145     const std::string& str, const std::string& suffix);  
00146  
00147 // Formats an int value as "%02d".  
00148 static std::string FormatIntWidth2(int value); // "%02d" for width == 2  
00149  
00150 // Formats an int value as "%X".  
00151 static std::string FormatHexInt(int value);  
00152  
00153 // Formats a byte as "%02X".  
00154 static std::string FormatByte(unsigned char value);  
00155  
00156 private:  
00157     String(); // Not meant to be instantiated.  
00158 }; // class String  
00159  
00160 // Gets the content of the stringstream's buffer as an std::string. Each '\0'  
00161 // character in the buffer is replaced with "\\0".  
00162 GTEST_API_ std::string StringStreamToString(::std::stringstream* stream);  
00163  
00164 } // namespace internal  
00165 } // namespace testing  
00166  
00167 #endif // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_STRING_H_
```

9.49 Dokumentacja pliku packages/Microsoft.gtest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-tuple.h

```
#include <utility>
```

Komponenty

- struct `std::tr1::gtest_internal::ByRef< T >`
- struct `std::tr1::gtest_internal::ByRef< T & >`
- struct `std::tr1::gtest_internal::AddRef< T >`
- struct `std::tr1::gtest_internal::AddRef< T & >`
- struct `std::tr1::gtest_internal::TupleElement< true, 0, GTEST_10_TUPLE_(T) >`
- struct `std::tr1::gtest_internal::TupleElement< true, 1, GTEST_10_TUPLE_(T) >`
- struct `std::tr1::gtest_internal::TupleElement< true, 2, GTEST_10_TUPLE_(T) >`
- struct `std::tr1::gtest_internal::TupleElement< true, 3, GTEST_10_TUPLE_(T) >`
- struct `std::tr1::gtest_internal::TupleElement< true, 4, GTEST_10_TUPLE_(T) >`
- struct `std::tr1::gtest_internal::TupleElement< true, 5, GTEST_10_TUPLE_(T) >`
- struct `std::tr1::gtest_internal::TupleElement< true, 6, GTEST_10_TUPLE_(T) >`
- struct `std::tr1::gtest_internal::TupleElement< true, 7, GTEST_10_TUPLE_(T) >`
- struct `std::tr1::gtest_internal::TupleElement< true, 8, GTEST_10_TUPLE_(T) >`
- struct `std::tr1::gtest_internal::TupleElement< true, 9, GTEST_10_TUPLE_(T) >`

- class `std::tr1::tuple<>`
- class `std::tr1::tuple<>`
- struct `std::tr1::tuple_size< GTEST_0_TUPLE_(T) >`
- struct `std::tr1::tuple_size< GTEST_1_TUPLE_(T) >`
- struct `std::tr1::tuple_size< GTEST_2_TUPLE_(T) >`
- struct `std::tr1::tuple_size< GTEST_3_TUPLE_(T) >`
- struct `std::tr1::tuple_size< GTEST_4_TUPLE_(T) >`
- struct `std::tr1::tuple_size< GTEST_5_TUPLE_(T) >`
- struct `std::tr1::tuple_size< GTEST_6_TUPLE_(T) >`
- struct `std::tr1::tuple_size< GTEST_7_TUPLE_(T) >`
- struct `std::tr1::tuple_size< GTEST_8_TUPLE_(T) >`
- struct `std::tr1::tuple_size< GTEST_9_TUPLE_(T) >`
- struct `std::tr1::tuple_size< GTEST_10_TUPLE_(T) >`
- struct `std::tr1::tuple_element< k, Tuple >`
- class `std::tr1::gtest_internal::Get< 0 >`
- class `std::tr1::gtest_internal::Get< 1 >`
- class `std::tr1::gtest_internal::Get< 2 >`
- class `std::tr1::gtest_internal::Get< 3 >`
- class `std::tr1::gtest_internal::Get< 4 >`
- class `std::tr1::gtest_internal::Get< 5 >`
- class `std::tr1::gtest_internal::Get< 6 >`
- class `std::tr1::gtest_internal::Get< 7 >`
- class `std::tr1::gtest_internal::Get< 8 >`
- class `std::tr1::gtest_internal::Get< 9 >`
- struct `std::tr1::gtest_internal::SameSizeTuplePrefixComparator< 0, 0 >`
- struct `std::tr1::gtest_internal::SameSizeTuplePrefixComparator< k, k >`

Przestrzenie nazw

- namespace `std`
- namespace `std::tr1`
- namespace `std::tr1::gtest_internal`

Definicje

- `#define GTEST_DECLARE_TUPLE_AS_FRIEND_`
- `#define GTEST_0_TUPLE_(T)`
- `#define GTEST_1_TUPLE_(T)`
- `#define GTEST_2_TUPLE_(T)`
- `#define GTEST_3_TUPLE_(T)`
- `#define GTEST_4_TUPLE_(T)`
- `#define GTEST_5_TUPLE_(T)`
- `#define GTEST_6_TUPLE_(T)`
- `#define GTEST_7_TUPLE_(T)`
- `#define GTEST_8_TUPLE_(T)`
- `#define GTEST_9_TUPLE_(T)`
- `#define GTEST_10_TUPLE_(T)`
- `#define GTEST_0_TYPENAMES_(T)`
- `#define GTEST_1_TYPENAMES_(T)`
- `#define GTEST_2_TYPENAMES_(T)`
- `#define GTEST_3_TYPENAMES_(T)`
- `#define GTEST_4_TYPENAMES_(T)`
- `#define GTEST_5_TYPENAMES_(T)`

- #define GTEST_6_TYPENAMES_(T)
- #define GTEST_7_TYPENAMES_(T)
- #define GTEST_8_TYPENAMES_(T)
- #define GTEST_9_TYPENAMES_(T)
- #define GTEST_10_TYPENAMES_(T)
- #define GTEST_BY_REF_(T)
- #define GTEST_ADD_REF_(T)
- #define GTEST_TUPLE_ELEMENT_(k, Tuple)

Funkcje

- template<GTEST_1_TYPENAMES_(T)>
class std::tr1::GTEST_1_TUPLE_(T)
- template<GTEST_2_TYPENAMES_(T)>
class std::tr1::GTEST_2_TUPLE_(T)
- template<GTEST_3_TYPENAMES_(T)>
class std::tr1::GTEST_3_TUPLE_(T)
- template<GTEST_4_TYPENAMES_(T)>
class std::tr1::GTEST_4_TUPLE_(T)
- template<GTEST_5_TYPENAMES_(T)>
class std::tr1::GTEST_5_TUPLE_(T)
- template<GTEST_6_TYPENAMES_(T)>
class std::tr1::GTEST_6_TUPLE_(T)
- template<GTEST_7_TYPENAMES_(T)>
class std::tr1::GTEST_7_TUPLE_(T)
- template<GTEST_8_TYPENAMES_(T)>
class std::tr1::GTEST_8_TUPLE_(T)
- template<GTEST_9_TYPENAMES_(T)>
class std::tr1::GTEST_9_TUPLE_(T)
- tuple std::tr1::make_tuple ()
- template<GTEST_1_TYPENAMES_(T)>
std::tr1::GTEST_1_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_2_TYPENAMES_(T)>
std::tr1::GTEST_2_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_3_TYPENAMES_(T)>
std::tr1::GTEST_3_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_4_TYPENAMES_(T)>
std::tr1::GTEST_4_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_5_TYPENAMES_(T)>
std::tr1::GTEST_5_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_6_TYPENAMES_(T)>
std::tr1::GTEST_6_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_7_TYPENAMES_(T)>
std::tr1::GTEST_7_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_8_TYPENAMES_(T)>
std::tr1::GTEST_8_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_9_TYPENAMES_(T)>
std::tr1::GTEST_9_TUPLE_(T) make_tuple(const T0 &f0)
- template<GTEST_10_TYPENAMES_(T)>
std::tr1::GTEST_10_TUPLE_(T) make_tuple(const T0 &f0)
- template<int k, GTEST_10_TYPENAMES_(T)>
std::tr1::GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(k, GTEST_10_TUPLE_(T))) get(GTEST_10_TUPLE_(T) &t)
- template<int k, GTEST_10_TYPENAMES_(T)>
std::tr1::GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(k, GTEST_10_TUPLE_(T))) get(const GTEST_10_TUPLE_(T) &t)

- template<GTEST_10_TYPENAMES_(T), GTEST_10_TYPENAMES_(U)>
bool std::tr1::operator==(const GTEST_10_TUPLE_(T)&t, const GTEST_10_TUPLE_(U)&u)
- template<GTEST_10_TYPENAMES_(T), GTEST_10_TYPENAMES_(U)>
bool std::tr1::operator!=(const GTEST_10_TUPLE_(T)&t, const GTEST_10_TUPLE_(U)&u)

Zmienne

- const T1 & std::tr1::f1
- const T1 const T2 & std::tr1::f2
- const T1 const T2 const T3 & std::tr1::f3
- const T1 const T2 const T3 const T4 & std::tr1::f4
- const T1 const T2 const T3 const T4 const T5 & std::tr1::f5
- const T1 const T2 const T3 const T4 const T5 const T6 & std::tr1::f6
- const T1 const T2 const T3 const T4 const T5 const T6 const T7 & std::tr1::f7
- const T1 const T2 const T3 const T4 const T5 const T6 const T7 const T8 & std::tr1::f8
- const T1 const T2 const T3 const T4 const T5 const T6 const T7 const T8 const T9 & std::tr1::f9

9.49.1 Dokumentacja definicji**9.49.1.1 GTEST_0_TUPLE_**

```
#define GTEST_0_TUPLE_(
    T)
```

Wartość:

```
tuple<>
```

9.49.1.2 GTEST_0_TYPENAMES_

```
#define GTEST_0_TYPENAMES_(
    T)
```

9.49.1.3 GTEST_10_TUPLE_

```
#define GTEST_10_TUPLE_(
    T)
```

Wartość:

```
tuple<T##0, T##1, T##2, T##3, T##4, T##5, T##6, \
      T##7, T##8, T##9>
```

9.49.1.4 GTEST_10_TYPENAMES_

```
#define GTEST_10_TYPENAMES_(
    T)
```

Wartość:

```
typename T##0, typename T##1, typename T##2, \
typename T##3, typename T##4, typename T##5, typename T##6, \
typename T##7, typename T##8, typename T##9
```

9.49.1.5 GTEST_1_TUPLE_

```
#define GTEST_1_TUPLE_(  
    T)
```

Wartość:

```
tuple<T##0, void, void, void, void, void, void, void, void>
```

9.49.1.6 GTEST_1_TYPENAMES_

```
#define GTEST_1_TYPENAMES_(  
    T)
```

Wartość:

```
typename T##0
```

9.49.1.7 GTEST_2_TUPLE_

```
#define GTEST_2_TUPLE_(  
    T)
```

Wartość:

```
tuple<T##0, T##1, void, void, void, void, void, void, void>
```

9.49.1.8 GTEST_2_TYPENAMES_

```
#define GTEST_2_TYPENAMES_(  
    T)
```

Wartość:

```
typename T##0, typename T##1
```

9.49.1.9 GTEST_3_TUPLE_

```
#define GTEST_3_TUPLE_(  
    T)
```

Wartość:

```
tuple<T##0, T##1, T##2, void, void, void, void, void, void>
```

9.49.1.10 GTEST_3_TYPENAMES_

```
#define GTEST_3_TYPENAMES_(  
    T)
```

Wartość:

```
typename T##0, typename T##1, typename T##2
```

9.49.1.11 GTEST_4_TUPLE_

```
#define GTEST_4_TUPLE_(
    T)
```

Wartość:

```
tuple<T##0, T##1, T##2, T##3, void, void, void, \
void, void, void>
```

9.49.1.12 GTEST_4_TYPENAMES_

```
#define GTEST_4_TYPENAMES_(
    T)
```

Wartość:

```
typename T##0, typename T##1, typename T##2, \
typename T##3
```

9.49.1.13 GTEST_5_TUPLE_

```
#define GTEST_5_TUPLE_(
    T)
```

Wartość:

```
tuple<T##0, T##1, T##2, T##3, T##4, void, void, \
void, void, void>
```

9.49.1.14 GTEST_5_TYPENAMES_

```
#define GTEST_5_TYPENAMES_(
    T)
```

Wartość:

```
typename T##0, typename T##1, typename T##2, \
typename T##3, typename T##4
```

9.49.1.15 GTEST_6_TUPLE_

```
#define GTEST_6_TUPLE_(
    T)
```

Wartość:

```
tuple<T##0, T##1, T##2, T##3, T##4, T##5, void, \
void, void, void>
```

9.49.1.16 GTEST_6_TYPENAMES_

```
#define GTEST_6_TYPENAMES_(
    T)
```

Wartość:

```
typename T##0, typename T##1, typename T##2, \
typename T##3, typename T##4, typename T##5
```

9.49.1.17 GTEST_7_TUPLE_

```
#define GTEST_7_TUPLE_(  
    T)
```

Wartość:

```
tuple<T##0, T##1, T##2, T##3, T##4, T##5, T##6, \  
void, void, void>
```

9.49.1.18 GTEST_7_TYPENAMES_

```
#define GTEST_7_TYPENAMES_(  
    T)
```

Wartość:

```
typename T##0, typename T##1, typename T##2, \  
typename T##3, typename T##4, typename T##5, typename T##6
```

9.49.1.19 GTEST_8_TUPLE_

```
#define GTEST_8_TUPLE_(  
    T)
```

Wartość:

```
tuple<T##0, T##1, T##2, T##3, T##4, T##5, T##6, T##7, \  
void, void>
```

9.49.1.20 GTEST_8_TYPENAMES_

```
#define GTEST_8_TYPENAMES_(  
    T)
```

Wartość:

```
typename T##0, typename T##1, typename T##2, \  
typename T##3, typename T##4, typename T##5, typename T##6, typename T##7
```

9.49.1.21 GTEST_9_TUPLE_

```
#define GTEST_9_TUPLE_(  
    T)
```

Wartość:

```
tuple<T##0, T##1, T##2, T##3, T##4, T##5, T##6, T##7, T##8, \  
void>
```

9.49.1.22 GTEST_9_TYPENAMES_

```
#define GTEST_9_TYPENAMES_(  
    T)
```

Wartość:

```
typename T##0, typename T##1, typename T##2, \  
typename T##3, typename T##4, typename T##5, typename T##6, typename T##7, typename T##8
```

9.49.1.23 GTEST_ADD_REF_

```
#define GTEST_ADD_REF_(  
    T)
```

Wartość:

```
typename ::std::tr1::gtest_internal::AddRef<T>::type
```

9.49.1.24 GTEST_BY_REF_

```
#define GTEST_BY_REF_(  
    T)
```

Wartość:

```
typename ::std::tr1::gtest_internal::ByRef<T>::type
```

9.49.1.25 GTEST_DECLARE_TUPLE_AS_FRIEND_

```
#define GTEST_DECLARE_TUPLE_AS_FRIEND_
```

Wartość:

```
template <GTEST_10_TYPENAMES_(U)> friend class tuple; \  
private:
```

9.49.1.26 GTEST_TUPLE_ELEMENT_

```
#define GTEST_TUPLE_ELEMENT_(  
    k,  
    Tuple)
```

Wartość:

```
typename tuple_element<k, Tuple >::type
```

9.50 gtest-tuple.h

[Idź do dokumentacji tego pliku.](#)

```
00001 // This file was GENERATED by command:  
00002 //     pump.py gtest-tuple.h.pump  
00003 // DO NOT EDIT BY HAND!!!  
00004  
00005 // Copyright 2009 Google Inc.  
00006 // All Rights Reserved.  
00007 //  
00008 // Redistribution and use in source and binary forms, with or without  
00009 // modification, are permitted provided that the following conditions are  
00010 // met:  
00011 //  
00012 //      * Redistributions of source code must retain the above copyright  
00013 // notice, this list of conditions and the following disclaimer.  
00014 //      * Redistributions in binary form must reproduce the above  
00015 // copyright notice, this list of conditions and the following disclaimer  
00016 // in the documentation and/or other materials provided with the  
00017 // distribution.  
00018 //      * Neither the name of Google Inc. nor the names of its  
00019 // contributors may be used to endorse or promote products derived from  
00020 // this software without specific prior written permission.  
00021 //
```

```

00022 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00023 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00024 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00025 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00026 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00027 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00028 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00029 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00030 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00031 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00032 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00033
00034
00035 // Implements a subset of TR1 tuple needed by Google Test and Google Mock.
00036
00037 // GOOGLETTEST_CM0001 DO NOT DELETE
00038
00039 #ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_TUPLE_H_
00040 #define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_TUPLE_H_
00041
00042 #include <utility> // For ::std::pair.
00043
00044 // The compiler used in Symbian has a bug that prevents us from declaring the
00045 // tuple template as a friend (it complains that tuple is redefined). This
00046 // bypasses the bug by declaring the members that should otherwise be
00047 // private as public.
00048 // Sun Studio versions < 12 also have the above bug.
00049 #if defined(__SYMBIAN32__) || (defined(__SUNPRO_CC) && __SUNPRO_CC < 0x590)
00050 # define GTEST_DECLARE_TUPLE_AS_FRIEND_ public:
00051 #else
00052 # define GTEST_DECLARE_TUPLE_AS_FRIEND_ \
00053     template <GTEST_10_TYPENAMES_(U)> friend class tuple; \
00054     private:
00055 #endif
00056
00057 // Visual Studio 2010, 2012, and 2013 define symbols in std::tr1 that conflict
00058 // with our own definitions. Therefore using our own tuple does not work on
00059 // those compilers.
00060 #if defined(_MSC_VER) && _MSC_VER >= 1600 /* 1600 is Visual Studio 2010 */
00061 # error "gtest's tuple doesn't compile on Visual Studio 2010 or later. \
00062 GTEST_USE_OWN_TR1_TUPLE must be set to 0 on those compilers."
00063 #endif
00064
00065 // GTEST_n_TUPLE_(T) is the type of an n-tuple.
00066 #define GTEST_0_TUPLE_(T) tuple<>
00067 #define GTEST_1_TUPLE_(T) tuple<T##0, void, void, void, void, void, void, \
00068     void, void, void>
00069 #define GTEST_2_TUPLE_(T) tuple<T##0, T##1, void, void, void, void, void, \
00070     void, void, void>
00071 #define GTEST_3_TUPLE_(T) tuple<T##0, T##1, T##2, void, void, void, void, \
00072     void, void, void>
00073 #define GTEST_4_TUPLE_(T) tuple<T##0, T##1, T##2, T##3, void, void, void, \
00074     void, void, void>
00075 #define GTEST_5_TUPLE_(T) tuple<T##0, T##1, T##2, T##3, T##4, void, void, \
00076     void, void, void>
00077 #define GTEST_6_TUPLE_(T) tuple<T##0, T##1, T##2, T##3, T##4, T##5, void, \
00078     void, void, void>
00079 #define GTEST_7_TUPLE_(T) tuple<T##0, T##1, T##2, T##3, T##4, T##5, T##6, \
00080     void, void, void>
00081 #define GTEST_8_TUPLE_(T) tuple<T##0, T##1, T##2, T##3, T##4, T##5, T##6, \
00082     T##7, void, void>
00083 #define GTEST_9_TUPLE_(T) tuple<T##0, T##1, T##2, T##3, T##4, T##5, T##6, \
00084     T##7, T##8, void>
00085 #define GTEST_10_TUPLE_(T) tuple<T##0, T##1, T##2, T##3, T##4, T##5, T##6, \
00086     T##7, T##8, T##9>
00087
00088 // GTEST_n_TYPENAMES_(T) declares a list of n typenames.
00089 #define GTEST_0_TYPENAMES_(T)
00090 #define GTEST_1_TYPENAMES_(T) typename T##0
00091 #define GTEST_2_TYPENAMES_(T) typename T##0, typename T##1
00092 #define GTEST_3_TYPENAMES_(T) typename T##0, typename T##1, typename T##2
00093 #define GTEST_4_TYPENAMES_(T) typename T##0, typename T##1, typename T##2, \
00094     typename T##3
00095 #define GTEST_5_TYPENAMES_(T) typename T##0, typename T##1, typename T##2, \
00096     typename T##3, typename T##4
00097 #define GTEST_6_TYPENAMES_(T) typename T##0, typename T##1, typename T##2, \
00098     typename T##3, typename T##4, typename T##5
00099 #define GTEST_7_TYPENAMES_(T) typename T##0, typename T##1, typename T##2, \
00100     typename T##3, typename T##4, typename T##5, typename T##6
00101 #define GTEST_8_TYPENAMES_(T) typename T##0, typename T##1, typename T##2, \
00102     typename T##3, typename T##4, typename T##5, typename T##6, typename T##7
00103 #define GTEST_9_TYPENAMES_(T) typename T##0, typename T##1, typename T##2, \
00104     typename T##3, typename T##4, typename T##5, typename T##6, \
00105     typename T##7, typename T##8
00106 #define GTEST_10_TYPENAMES_(T) typename T##0, typename T##1, typename T##2, \
00107     typename T##3, typename T##4, typename T##5, typename T##6, \
00108     typename T##7, typename T##8, typename T##9

```

```
00109
00110 // In theory, defining stuff in the ::std namespace is undefined
00111 // behavior. We can do this as we are playing the role of a standard
00112 // library vendor.
00113 namespace std {
00114 namespace tr1 {
00115
00116 template <typename T0 = void, typename T1 = void, typename T2 = void,
00117     typename T3 = void, typename T4 = void, typename T5 = void,
00118     typename T6 = void, typename T7 = void, typename T8 = void,
00119     typename T9 = void>
00120 class tuple;
00121
00122 // Anything in namespace gtest_internal is Google Test's INTERNAL
00123 // IMPLEMENTATION DETAIL and MUST NOT BE USED DIRECTLY in user code.
00124 namespace gtest_internal {
00125
00126 // ByRef<T>::type is T if T is a reference; otherwise it's const T&.
00127 template <typename T>
00128 struct ByRef { typedef const T& type; }; // NOLINT
00129 template <typename T>
00130 struct ByRef<T&> { typedef T& type; }; // NOLINT
00131
00132 // A handy wrapper for ByRef.
00133 #define GTEST_BY_REF_(T) typename ::std::tr1::gtest_internal::ByRef<T>::type
00134
00135 // AddRef<T>::type is T if T is a reference; otherwise it's T&. This
00136 // is the same as tr1::add_reference<T>::type.
00137 template <typename T>
00138 struct AddRef { typedef T& type; }; // NOLINT
00139 template <typename T>
00140 struct AddRef<T&> { typedef T& type; }; // NOLINT
00141
00142 // A handy wrapper for AddRef.
00143 #define GTEST_ADD_REF_(T) typename ::std::tr1::gtest_internal::AddRef<T>::type
00144
00145 // A helper for implementing get<k>().
00146 template <int k> class Get;
00147
00148 // A helper for implementing tuple_element<k, T>. kIndexValid is true
00149 // iff k < the number of fields in tuple type T.
00150 template <bool kIndexValid, int kIndex, class Tuple>
00151 struct TupleElement;
00152
00153 template <GTEST_10_TYPENAMES_(T)>
00154 struct TupleElement<true, 0, GTEST_10_TUPLE_(T) > {
00155     typedef T0 type;
00156 };
00157
00158 template <GTEST_10_TYPENAMES_(T)>
00159 struct TupleElement<true, 1, GTEST_10_TUPLE_(T) > {
00160     typedef T1 type;
00161 };
00162
00163 template <GTEST_10_TYPENAMES_(T)>
00164 struct TupleElement<true, 2, GTEST_10_TUPLE_(T) > {
00165     typedef T2 type;
00166 };
00167
00168 template <GTEST_10_TYPENAMES_(T)>
00169 struct TupleElement<true, 3, GTEST_10_TUPLE_(T) > {
00170     typedef T3 type;
00171 };
00172
00173 template <GTEST_10_TYPENAMES_(T)>
00174 struct TupleElement<true, 4, GTEST_10_TUPLE_(T) > {
00175     typedef T4 type;
00176 };
00177
00178 template <GTEST_10_TYPENAMES_(T)>
00179 struct TupleElement<true, 5, GTEST_10_TUPLE_(T) > {
00180     typedef T5 type;
00181 };
00182
00183 template <GTEST_10_TYPENAMES_(T)>
00184 struct TupleElement<true, 6, GTEST_10_TUPLE_(T) > {
00185     typedef T6 type;
00186 };
00187
00188 template <GTEST_10_TYPENAMES_(T)>
00189 struct TupleElement<true, 7, GTEST_10_TUPLE_(T) > {
00190     typedef T7 type;
00191 };
00192
00193 template <GTEST_10_TYPENAMES_(T)>
00194 struct TupleElement<true, 8, GTEST_10_TUPLE_(T) > {
00195     typedef T8 type;
```

```
00196 };
00197
00198 template <GTEST_10_TYPENAMES_(T)>
00199 struct TupleElement<true, 9, GTEST_10_TUPLE_(T) > {
00200     typedef T9 type;
00201 };
00202
00203 } // namespace gtest_internal
00204
00205 template <>
00206 class tuple<> {
00207     public:
00208     tuple() {}
00209     tuple(const tuple& /* t */) {}
00210     tuple& operator=(const tuple& /* t */) { return *this; }
00211 };
00212
00213 template <GTEST_1_TYPENAMES_(T)>
00214 class GTEST_1_TUPLE_(T) {
00215     public:
00216     template <int k> friend class gtest_internal::Get;
00217
00218     tuple() : f0_() {}
00219
00220     explicit tuple(GTEST_BY_REF_(T0) f0) : f0_(f0) {}
00221
00222     tuple(const tuple& t) : f0_(t.f0_) {}
00223
00224     template <GTEST_1_TYPENAMES_(U)>
00225     tuple(const GTEST_1_TUPLE_(U)& t) : f0_(t.f0_) {}
00226
00227     tuple& operator=(const tuple& t) { return CopyFrom(t); }
00228
00229     template <GTEST_1_TYPENAMES_(U)>
00230     tuple& operator=(const GTEST_1_TUPLE_(U)& t) {
00231         return CopyFrom(t);
00232     }
00233
00234     GTEST_DECLARE_TUPLE_AS_FRIEND_
00235
00236     template <GTEST_1_TYPENAMES_(U)>
00237     tuple& CopyFrom(const GTEST_1_TUPLE_(U)& t) {
00238         f0_ = t.f0_;
00239         return *this;
00240     }
00241
00242     T0 f0_;
00243 };
00244
00245 template <GTEST_2_TYPENAMES_(T)>
00246 class GTEST_2_TUPLE_(T) {
00247     public:
00248     template <int k> friend class gtest_internal::Get;
00249
00250     tuple() : f0_(), f1_() {}
00251
00252     explicit tuple(GTEST_BY_REF_(T0) f0, GTEST_BY_REF_(T1) f1) : f0_(f0),
00253         f1_(f1) {}
00254
00255     tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_) {}
00256
00257     template <GTEST_2_TYPENAMES_(U)>
00258     tuple(const GTEST_2_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_) {}
00259     template <typename U0, typename U1>
00260     tuple(const ::std::pair<U0, U1>& p) : f0_(p.first), f1_(p.second) {}
00261
00262     tuple& operator=(const tuple& t) { return CopyFrom(t); }
00263
00264     template <GTEST_2_TYPENAMES_(U)>
00265     tuple& operator=(const GTEST_2_TUPLE_(U)& t) {
00266         return CopyFrom(t);
00267     }
00268     template <typename U0, typename U1>
00269     tuple& operator=(const ::std::pair<U0, U1>& p) {
00270         f0_ = p.first;
00271         f1_ = p.second;
00272         return *this;
00273     }
00274
00275     GTEST_DECLARE_TUPLE_AS_FRIEND_
00276
00277     template <GTEST_2_TYPENAMES_(U)>
00278     tuple& CopyFrom(const GTEST_2_TUPLE_(U)& t) {
00279         f0_ = t.f0_;
00280         f1_ = t.f1_;
00281         return *this;
00282     }
```

```

00283
00284     T0 f0_;
00285     T1 f1_;
00286 };
00287
00288 template <GTEST_3_TYPENAMES_(T)>
00289 class GTEST_3_TUPLE_(T) {
00290 public:
00291     template <int k> friend class gtest_internal::Get;
00292
00293     tuple() : f0_(()), f1_(()), f2_(()) {}
00294
00295     explicit tuple(GTEST_BY_REF_(T0) f0, GTEST_BY_REF_(T1) f1,
00296                     GTEST_BY_REF_(T2) f2) : f0_(f0), f1_(f1), f2_(f2) {}
00297
00298     tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_) {}
00299
00300     template <GTEST_3_TYPENAMES_(U)>
00301     tuple(const GTEST_3_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_) {}
00302
00303     tuple& operator=(const tuple& t) { return CopyFrom(t); }
00304
00305     template <GTEST_3_TYPENAMES_(U)>
00306     tuple& operator=(const GTEST_3_TUPLE_(U)& t) {
00307         return CopyFrom(t);
00308     }
00309
00310     GTEST_DECLARE_TUPLE_AS_FRIEND_
00311
00312     template <GTEST_3_TYPENAMES_(U)>
00313     tuple& CopyFrom(const GTEST_3_TUPLE_(U)& t) {
00314         f0_ = t.f0_;
00315         f1_ = t.f1_;
00316         f2_ = t.f2_;
00317         return *this;
00318     }
00319
00320     T0 f0_;
00321     T1 f1_;
00322     T2 f2_;
00323 };
00324
00325 template <GTEST_4_TYPENAMES_(T)>
00326 class GTEST_4_TUPLE_(T) {
00327 public:
00328     template <int k> friend class gtest_internal::Get;
00329
00330     tuple() : f0_(()), f1_(()), f2_(()), f3_(()) {}
00331
00332     explicit tuple(GTEST_BY_REF_(T0) f0, GTEST_BY_REF_(T1) f1,
00333                     GTEST_BY_REF_(T2) f2, GTEST_BY_REF_(T3) f3) : f0_(f0), f1_(f1), f2_(f2),
00334                     f3_(f3) {}
00335
00336     tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_), f3_(t.f3_) {}
00337
00338     template <GTEST_4_TYPENAMES_(U)>
00339     tuple(const GTEST_4_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_),
00340                     f3_(t.f3_) {}
00341
00342     tuple& operator=(const tuple& t) { return CopyFrom(t); }
00343
00344     template <GTEST_4_TYPENAMES_(U)>
00345     tuple& operator=(const GTEST_4_TUPLE_(U)& t) {
00346         return CopyFrom(t);
00347     }
00348
00349     GTEST_DECLARE_TUPLE_AS_FRIEND_
00350
00351     template <GTEST_4_TYPENAMES_(U)>
00352     tuple& CopyFrom(const GTEST_4_TUPLE_(U)& t) {
00353         f0_ = t.f0_;
00354         f1_ = t.f1_;
00355         f2_ = t.f2_;
00356         f3_ = t.f3_;
00357         return *this;
00358     }
00359
00360     T0 f0_;
00361     T1 f1_;
00362     T2 f2_;
00363     T3 f3_;
00364 };
00365
00366 template <GTEST_5_TYPENAMES_(T)>
00367 class GTEST_5_TUPLE_(T) {
00368 public:
00369     template <int k> friend class gtest_internal::Get;

```

```
00370
00371     tuple() : f0_(), f1_(), f2_(), f3_(), f4_() {}
00372
00373     explicit tuple(GTEST_BY_REF_(T0) f0, GTEST_BY_REF_(T1) f1,
00374         GTEST_BY_REF_(T2) f2, GTEST_BY_REF_(T3) f3,
00375         GTEST_BY_REF_(T4) f4) : f0_(f0), f1_(f1), f2_(f2), f3_(f3), f4_(f4) {}
00376
00377     tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_), f3_(t.f3_),
00378         f4_(t.f4_) {}
00379
00380     template <GTEST_5_TYPENAMES_(U)>
00381     tuple(const GTEST_5_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_),
00382         f3_(t.f3_), f4_(t.f4_) {}
00383
00384     tuple& operator=(const tuple& t) { return CopyFrom(t); }
00385
00386     template <GTEST_5_TYPENAMES_(U)>
00387     tuple& operator=(const GTEST_5_TUPLE_(U)& t) {
00388         return CopyFrom(t);
00389     }
00390
00391     GTEST_DECLARE_TUPLE_AS_FRIEND_
00392
00393     template <GTEST_5_TYPENAMES_(U)>
00394     tuple& CopyFrom(const GTEST_5_TUPLE_(U)& t) {
00395         f0_ = t.f0_;
00396         f1_ = t.f1_;
00397         f2_ = t.f2_;
00398         f3_ = t.f3_;
00399         f4_ = t.f4_;
00400         return *this;
00401     }
00402
00403     T0 f0_;
00404     T1 f1_;
00405     T2 f2_;
00406     T3 f3_;
00407     T4 f4_;
00408 };
00409
00410     template <GTEST_6_TYPENAMES_(T)>
00411     class GTEST_6_TUPLE_(T) {
00412         public:
00413             template <int k> friend class gtest_internal::Get;
00414
00415             tuple() : f0_(), f1_(), f2_(), f3_(), f4_(), f5_() {}
00416
00417             explicit tuple(GTEST_BY_REF_(T0) f0, GTEST_BY_REF_(T1) f1,
00418                 GTEST_BY_REF_(T2) f2, GTEST_BY_REF_(T3) f3, GTEST_BY_REF_(T4) f4,
00419                 GTEST_BY_REF_(T5) f5) : f0_(f0), f1_(f1), f2_(f2), f3_(f3), f4_(f4),
00420                 f5_(f5) {}
00421
00422             tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_), f3_(t.f3_),
00423                 f4_(t.f4_), f5_(t.f5_) {}
00424
00425             template <GTEST_6_TYPENAMES_(U)>
00426             tuple(const GTEST_6_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_),
00427                 f3_(t.f3_), f4_(t.f4_), f5_(t.f5_) {}
00428
00429             tuple& operator=(const tuple& t) { return CopyFrom(t); }
00430
00431             template <GTEST_6_TYPENAMES_(U)>
00432             tuple& operator=(const GTEST_6_TUPLE_(U)& t) {
00433                 return CopyFrom(t);
00434             }
00435
00436             GTEST_DECLARE_TUPLE_AS_FRIEND_
00437
00438             template <GTEST_6_TYPENAMES_(U)>
00439             tuple& CopyFrom(const GTEST_6_TUPLE_(U)& t) {
00440                 f0_ = t.f0_;
00441                 f1_ = t.f1_;
00442                 f2_ = t.f2_;
00443                 f3_ = t.f3_;
00444                 f4_ = t.f4_;
00445                 f5_ = t.f5_;
00446                 return *this;
00447             }
00448
00449             T0 f0_;
00450             T1 f1_;
00451             T2 f2_;
00452             T3 f3_;
00453             T4 f4_;
00454             T5 f5_;
00455 };
00456
```

```
00457 template <GTEST_7_TYPENAMES_(T)>
00458 class GTEST_7_TUPLE_(T) {
00459 public:
00460     template <int k> friend class gtest_internal::Get;
00461     tuple() : f0_(), f1_(), f2_(), f3_(), f4_(), f5_(), f6_() {}
00463
00464     explicit tuple(GTEST_BY_REF_(T0) f0, GTEST_BY_REF_(T1) f1,
00465         GTEST_BY_REF_(T2) f2, GTEST_BY_REF_(T3) f3, GTEST_BY_REF_(T4) f4,
00466         GTEST_BY_REF_(T5) f5, GTEST_BY_REF_(T6) f6) : f0_(f0), f1_(f1), f2_(f2),
00467             f3_(f3), f4_(f4), f5_(f5), f6_(f6) {}
00468
00469     tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_), f3_(t.f3_),
00470         f4_(t.f4_), f5_(t.f5_), f6_(t.f6_) {}
00471
00472     template <GTEST_7_TYPENAMES_(U)>
00473     tuple(const GTEST_7_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_),
00474         f3_(t.f3_), f4_(t.f4_), f5_(t.f5_), f6_(t.f6_) {}
00475
00476     tuple& operator=(const tuple& t) { return CopyFrom(t); }
00477
00478     template <GTEST_7_TYPENAMES_(U)>
00479     tuple& operator=(const GTEST_7_TUPLE_(U)& t) {
00480         return CopyFrom(t);
00481     }
00482
00483     GTEST_DECLARE_TUPLE_AS_FRIEND_
00484
00485     template <GTEST_7_TYPENAMES_(U)>
00486     tuple& CopyFrom(const GTEST_7_TUPLE_(U)& t) {
00487         f0_ = t.f0_;
00488         f1_ = t.f1_;
00489         f2_ = t.f2_;
00490         f3_ = t.f3_;
00491         f4_ = t.f4_;
00492         f5_ = t.f5_;
00493         f6_ = t.f6_;
00494         return *this;
00495     }
00496
00497     T0 f0_;
00498     T1 f1_;
00499     T2 f2_;
00500     T3 f3_;
00501     T4 f4_;
00502     T5 f5_;
00503     T6 f6_;
00504 };
00505
00506 template <GTEST_8_TYPENAMES_(T)>
00507 class GTEST_8_TUPLE_(T) {
00508 public:
00509     template <int k> friend class gtest_internal::Get;
00510     tuple() : f0_(), f1_(), f2_(), f3_(), f4_(), f5_(), f6_(), f7_() {}
00512
00513     explicit tuple(GTEST_BY_REF_(T0) f0, GTEST_BY_REF_(T1) f1,
00514         GTEST_BY_REF_(T2) f2, GTEST_BY_REF_(T3) f3, GTEST_BY_REF_(T4) f4,
00515         GTEST_BY_REF_(T5) f5, GTEST_BY_REF_(T6) f6,
00516         GTEST_BY_REF_(T7) f7) : f0_(f0), f1_(f1), f2_(f2), f3_(f3), f4_(f4),
00517             f5_(f5), f6_(f6), f7_(f7) {}
00518
00519     tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_), f3_(t.f3_),
00520         f4_(t.f4_), f5_(t.f5_), f6_(t.f6_), f7_(t.f7_) {}
00521
00522     template <GTEST_8_TYPENAMES_(U)>
00523     tuple(const GTEST_8_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_),
00524         f3_(t.f3_), f4_(t.f4_), f5_(t.f5_), f6_(t.f6_), f7_(t.f7_) {}
00525
00526     tuple& operator=(const tuple& t) { return CopyFrom(t); }
00527
00528     template <GTEST_8_TYPENAMES_(U)>
00529     tuple& operator=(const GTEST_8_TUPLE_(U)& t) {
00530         return CopyFrom(t);
00531     }
00532
00533     GTEST_DECLARE_TUPLE_AS_FRIEND_
00534
00535     template <GTEST_8_TYPENAMES_(U)>
00536     tuple& CopyFrom(const GTEST_8_TUPLE_(U)& t) {
00537         f0_ = t.f0_;
00538         f1_ = t.f1_;
00539         f2_ = t.f2_;
00540         f3_ = t.f3_;
00541         f4_ = t.f4_;
00542         f5_ = t.f5_;
00543         f6_ = t.f6_;
```

```

00544     f7_ = t.f7_;
00545     return *this;
00546 }
00547
00548 T0 f0_;
00549 T1 f1_;
00550 T2 f2_;
00551 T3 f3_;
00552 T4 f4_;
00553 T5 f5_;
00554 T6 f6_;
00555 T7 f7_;
00556 };
00557
00558 template <GTEST_9_TYPENAMES_(T)>
00559 class GTEST_9_TUPLE_(T) {
00560 public:
00561     template <int k> friend class gtest_internal::Get;
00562
00563     tuple() : f0_(), f1_(), f2_(), f3_(), f4_(), f5_(), f6_(), f7_(), f8_() {}
00564
00565     explicit tuple(GTEST_BY_REF_(T0) f0, GTEST_BY_REF_(T1) f1,
00566                     GTEST_BY_REF_(T2) f2, GTEST_BY_REF_(T3) f3, GTEST_BY_REF_(T4) f4,
00567                     GTEST_BY_REF_(T5) f5, GTEST_BY_REF_(T6) f6, GTEST_BY_REF_(T7) f7,
00568                     GTEST_BY_REF_(T8) f8) : f0_(f0), f1_(f1), f2_(f2), f3_(f3), f4_(f4),
00569                     f5_(f5), f6_(f6), f7_(f7), f8_(f8) {}
00570
00571     tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_), f3_(t.f3_),
00572                     f4_(t.f4_), f5_(t.f5_), f6_(t.f6_), f7_(t.f7_), f8_(t.f8_) {}
00573
00574     template <GTEST_9_TYPENAMES_(U)>
00575     tuple(const GTEST_9_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_),
00576                     f3_(t.f3_), f4_(t.f4_), f5_(t.f5_), f6_(t.f6_), f7_(t.f7_), f8_(t.f8_) {}
00577
00578     tuple& operator=(const tuple& t) { return CopyFrom(t); }
00579
00580     template <GTEST_9_TYPENAMES_(U)>
00581     tuple& operator=(const GTEST_9_TUPLE_(U)& t) {
00582         return CopyFrom(t);
00583     }
00584
00585     GTEST_DECLARE_TUPLE_AS_FRIEND_
00586
00587     template <GTEST_9_TYPENAMES_(U)>
00588     tuple& CopyFrom(const GTEST_9_TUPLE_(U)& t) {
00589         f0_ = t.f0_;
00590         f1_ = t.f1_;
00591         f2_ = t.f2_;
00592         f3_ = t.f3_;
00593         f4_ = t.f4_;
00594         f5_ = t.f5_;
00595         f6_ = t.f6_;
00596         f7_ = t.f7_;
00597         f8_ = t.f8_;
00598         return *this;
00599     }
00600
00601 T0 f0_;
00602 T1 f1_;
00603 T2 f2_;
00604 T3 f3_;
00605 T4 f4_;
00606 T5 f5_;
00607 T6 f6_;
00608 T7 f7_;
00609 T8 f8_;
00610 };
00611
00612 template <GTEST_10_TYPENAMES_(T)>
00613 class tuple {
00614 public:
00615     template <int k> friend class gtest_internal::Get;
00616
00617     tuple() : f0_(), f1_(), f2_(), f3_(), f4_(), f5_(), f6_(), f7_(), f8_(),
00618                     f9_() {}
00619
00620     explicit tuple(GTEST_BY_REF_(T0) f0, GTEST_BY_REF_(T1) f1,
00621                     GTEST_BY_REF_(T2) f2, GTEST_BY_REF_(T3) f3, GTEST_BY_REF_(T4) f4,
00622                     GTEST_BY_REF_(T5) f5, GTEST_BY_REF_(T6) f6, GTEST_BY_REF_(T7) f7,
00623                     GTEST_BY_REF_(T8) f8, GTEST_BY_REF_(T9) f9) : f0_(f0), f1_(f1), f2_(f2),
00624                     f3_(f3), f4_(f4), f5_(f5), f6_(f6), f7_(f7), f8_(f8), f9_(f9) {}
00625
00626     tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_), f3_(t.f3_),
00627                     f4_(t.f4_), f5_(t.f5_), f6_(t.f6_), f7_(t.f7_), f8_(t.f8_), f9_(t.f9_) {}
00628
00629     template <GTEST_10_TYPENAMES_(U)>
00630     tuple(const GTEST_10_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_),
```

```

00631     f3_(t.f3_), f4_(t.f4_), f5_(t.f5_), f6_(t.f6_), f7_(t.f7_), f8_(t.f8_),
00632     f9_(t.f9_) {}
00633
00634     tuple& operator=(const tuple& t) { return CopyFrom(t); }
00635
00636     template <GTEST_10_TYPENAMES_(U)>
00637     tuple& operator=(const GTEST_10_TUPLE_(U)& t) {
00638         return CopyFrom(t);
00639     }
00640
00641     GTEST_DECLARE_TUPLE_AS_FRIEND_
00642
00643     template <GTEST_10_TYPENAMES_(U)>
00644     tuple& CopyFrom(const GTEST_10_TUPLE_(U)& t) {
00645         f0_ = t.f0_;
00646         f1_ = t.f1_;
00647         f2_ = t.f2_;
00648         f3_ = t.f3_;
00649         f4_ = t.f4_;
00650         f5_ = t.f5_;
00651         f6_ = t.f6_;
00652         f7_ = t.f7_;
00653         f8_ = t.f8_;
00654         f9_ = t.f9_;
00655         return *this;
00656     }
00657
00658     T0 f0_;
00659     T1 f1_;
00660     T2 f2_;
00661     T3 f3_;
00662     T4 f4_;
00663     T5 f5_;
00664     T6 f6_;
00665     T7 f7_;
00666     T8 f8_;
00667     T9 f9_;
00668 };
00669
00670 // 6.1.3.2 Tuple creation functions.
00671
00672 // Known limitations: we don't support passing an
00673 // std::tr1::reference_wrapper<T> to make_tuple(). And we don't
00674 // implement tie().
00675
00676 inline tuple<> make_tuple() { return tuple<>(); }
00677
00678 template <GTEST_1_TYPENAMES_(T)>
00679 inline GTEST_1_TUPLE_(T) make_tuple(const T0& f0) {
00680     return GTEST_1_TUPLE_(T)(f0);
00681 }
00682
00683 template <GTEST_2_TYPENAMES_(T)>
00684 inline GTEST_2_TUPLE_(T) make_tuple(const T0& f0, const T1& f1) {
00685     return GTEST_2_TUPLE_(T)(f0, f1);
00686 }
00687
00688 template <GTEST_3_TYPENAMES_(T)>
00689 inline GTEST_3_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2) {
00690     return GTEST_3_TUPLE_(T)(f0, f1, f2);
00691 }
00692
00693 template <GTEST_4_TYPENAMES_(T)>
00694 inline GTEST_4_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2,
00695     const T3& f3) {
00696     return GTEST_4_TUPLE_(T)(f0, f1, f2, f3);
00697 }
00698
00699 template <GTEST_5_TYPENAMES_(T)>
00700 inline GTEST_5_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2,
00701     const T3& f3, const T4& f4) {
00702     return GTEST_5_TUPLE_(T)(f0, f1, f2, f3, f4);
00703 }
00704
00705 template <GTEST_6_TYPENAMES_(T)>
00706 inline GTEST_6_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2,
00707     const T3& f3, const T4& f4, const T5& f5) {
00708     return GTEST_6_TUPLE_(T)(f0, f1, f2, f3, f4, f5);
00709 }
00710
00711 template <GTEST_7_TYPENAMES_(T)>
00712 inline GTEST_7_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2,
00713     const T3& f3, const T4& f4, const T5& f5, const T6& f6) {
00714     return GTEST_7_TUPLE_(T)(f0, f1, f2, f3, f4, f5, f6);
00715 }
00716
00717 template <GTEST_8_TYPENAMES_(T)>

```

```
00718 inline GTEST_8_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2,
00719     const T3& f3, const T4& f4, const T5& f5, const T6& f6, const T7& f7) {
00720     return GTEST_8_TUPLE_(T)(f0, f1, f2, f3, f4, f5, f6, f7);
00721 }
00722
00723 template <GTEST_9_TYPENAMES_(T)>
00724 inline GTEST_9_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2,
00725     const T3& f3, const T4& f4, const T5& f5, const T6& f6, const T7& f7,
00726     const T8& f8) {
00727     return GTEST_9_TUPLE_(T)(f0, f1, f2, f3, f4, f5, f6, f7, f8);
00728 }
00729
00730 template <GTEST_10_TYPENAMES_(T)>
00731 inline GTEST_10_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2,
00732     const T3& f3, const T4& f4, const T5& f5, const T6& f6, const T7& f7,
00733     const T8& f8, const T9& f9) {
00734     return GTEST_10_TUPLE_(T)(f0, f1, f2, f3, f4, f5, f6, f7, f8, f9);
00735 }
00736
00737 // 6.1.3.3 Tuple helper classes.
00738
00739 template <typename Tuple> struct tuple_size;
00740
00741 template <GTEST_0_TYPENAMES_(T)>
00742 struct tuple_size<GTEST_0_TUPLE_(T) > {
00743     static const int value = 0;
00744 };
00745
00746 template <GTEST_1_TYPENAMES_(T)>
00747 struct tuple_size<GTEST_1_TUPLE_(T) > {
00748     static const int value = 1;
00749 };
00750
00751 template <GTEST_2_TYPENAMES_(T)>
00752 struct tuple_size<GTEST_2_TUPLE_(T) > {
00753     static const int value = 2;
00754 };
00755
00756 template <GTEST_3_TYPENAMES_(T)>
00757 struct tuple_size<GTEST_3_TUPLE_(T) > {
00758     static const int value = 3;
00759 };
00760
00761 template <GTEST_4_TYPENAMES_(T)>
00762 struct tuple_size<GTEST_4_TUPLE_(T) > {
00763     static const int value = 4;
00764 };
00765
00766 template <GTEST_5_TYPENAMES_(T)>
00767 struct tuple_size<GTEST_5_TUPLE_(T) > {
00768     static const int value = 5;
00769 };
00770
00771 template <GTEST_6_TYPENAMES_(T)>
00772 struct tuple_size<GTEST_6_TUPLE_(T) > {
00773     static const int value = 6;
00774 };
00775
00776 template <GTEST_7_TYPENAMES_(T)>
00777 struct tuple_size<GTEST_7_TUPLE_(T) > {
00778     static const int value = 7;
00779 };
00780
00781 template <GTEST_8_TYPENAMES_(T)>
00782 struct tuple_size<GTEST_8_TUPLE_(T) > {
00783     static const int value = 8;
00784 };
00785
00786 template <GTEST_9_TYPENAMES_(T)>
00787 struct tuple_size<GTEST_9_TUPLE_(T) > {
00788     static const int value = 9;
00789 };
00790
00791 template <GTEST_10_TYPENAMES_(T)>
00792 struct tuple_size<GTEST_10_TUPLE_(T) > {
00793     static const int value = 10;
00794 };
00795
00796 template <int k, class Tuple>
00797 struct tuple_element {
00798     typedef typename gtest_internal::TupleElement<
00799         k < (tuple_size<Tuple>::value), k, Tuple>::type type;
00800 };
00801
00802 #define GTEST_TUPLE_ELEMENT_(k, Tuple) typename tuple_element<k, Tuple >::type
00803
00804 // 6.1.3.4 Element access.
```

```
00805 namespace gtest_internal {
00806     template <>
00807     class Get<0> {
00808         public:
00809             template <class Tuple>
00810             static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(0, Tuple))
00811             Field(Tuple& t) { return t.f0_; } // NOLINT
00812
00813             template <class Tuple>
00814             static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(0, Tuple))
00815             ConstField(const Tuple& t) { return t.f0_; }
00816         };
00817
00818         template <>
00819         class Get<1> {
00820             public:
00821                 template <class Tuple>
00822                 static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(1, Tuple))
00823                 Field(Tuple& t) { return t.f1_; } // NOLINT
00824
00825                 template <class Tuple>
00826                 static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(1, Tuple))
00827                 ConstField(const Tuple& t) { return t.f1_; }
00828             };
00829
00830         template <>
00831         class Get<2> {
00832             public:
00833                 template <class Tuple>
00834                 static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(2, Tuple))
00835                 Field(Tuple& t) { return t.f2_; } // NOLINT
00836
00837                 template <class Tuple>
00838                 static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(2, Tuple))
00839                 ConstField(const Tuple& t) { return t.f2_; }
00840             };
00841
00842         template <>
00843         class Get<3> {
00844             public:
00845                 template <class Tuple>
00846                 static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(3, Tuple))
00847                 Field(Tuple& t) { return t.f3_; } // NOLINT
00848
00849                 template <class Tuple>
00850                 static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(3, Tuple))
00851                 ConstField(const Tuple& t) { return t.f3_; }
00852             };
00853
00854         template <>
00855         class Get<4> {
00856             public:
00857                 template <class Tuple>
00858                 static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(4, Tuple))
00859                 Field(Tuple& t) { return t.f4_; } // NOLINT
00860
00861                 template <class Tuple>
00862                 static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(4, Tuple))
00863                 ConstField(const Tuple& t) { return t.f4_; }
00864             };
00865
00866         template <>
00867         class Get<5> {
00868             public:
00869                 template <class Tuple>
00870                 static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(5, Tuple))
00871                 Field(Tuple& t) { return t.f5_; } // NOLINT
00872
00873                 template <class Tuple>
00874                 static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(5, Tuple))
00875                 ConstField(const Tuple& t) { return t.f5_; }
00876             };
00877
00878         template <>
00879         class Get<6> {
00880             public:
00881                 template <class Tuple>
00882                 static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(6, Tuple))
00883                 Field(Tuple& t) { return t.f6_; } // NOLINT
00884
00885                 template <class Tuple>
00886                 static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(6, Tuple))
00887                 ConstField(const Tuple& t) { return t.f6_; }
00888             };
00889
00890         };
00891 }
```

```
00892 template <>
00893 class Get<7> {
00894 public:
00895     template <class Tuple>
00896     static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(7, Tuple))
00897     Field(Tuple& t) { return t.f7_; } // NOLINT
00898
00899     template <class Tuple>
00900     static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(7, Tuple))
00901     ConstField(const Tuple& t) { return t.f7_; }
00902 };
00903
00904 template <>
00905 class Get<8> {
00906 public:
00907     template <class Tuple>
00908     static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(8, Tuple))
00909     Field(Tuple& t) { return t.f8_; } // NOLINT
00910
00911     template <class Tuple>
00912     static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(8, Tuple))
00913     ConstField(const Tuple& t) { return t.f8_; }
00914 };
00915
00916 template <>
00917 class Get<9> {
00918 public:
00919     template <class Tuple>
00920     static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(9, Tuple))
00921     Field(Tuple& t) { return t.f9_; } // NOLINT
00922
00923     template <class Tuple>
00924     static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(9, Tuple))
00925     ConstField(const Tuple& t) { return t.f9_; }
00926 };
00927
00928 } // namespace gtest_internal
00929
00930 template <int k, GTEST_10_TYPENAMES_(T)>
00931 GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(k, GTEST_10_TUPLE_(T)))
00932 get(GTEST_10_TUPLE_(T)& t) {
00933     return gtest_internal::Get<k>::Field(t);
00934 }
00935
00936 template <int k, GTEST_10_TYPENAMES_(T)>
00937 GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(k, GTEST_10_TUPLE_(T)))
00938 get(const GTEST_10_TUPLE_(T)& t) {
00939     return gtest_internal::Get<k>::ConstField(t);
00940 }
00941
00942 // 6.1.3.5 Relational operators
00943
00944 // We only implement == and !=, as we don't have a need for the rest yet.
00945
00946 namespace gtest_internal {
00947
00948 // SameSizeTuplePrefixComparator<k, k>::Eq(t1, t2) returns true if the
00949 // first k fields of t1 equals the first k fields of t2.
00950 // SameSizeTuplePrefixComparator(k1, k2) would be a compiler error if
00951 // k1 != k2.
00952 template <int kSize1, int kSize2>
00953 struct SameSizeTuplePrefixComparator;
00954
00955 template <>
00956 struct SameSizeTuplePrefixComparator<0, 0> {
00957     template <class Tuple1, class Tuple2>
00958     static bool Eq(const Tuple1& /* t1 */, const Tuple2& /* t2 */) {
00959         return true;
00960     }
00961 };
00962
00963 template <int k>
00964 struct SameSizeTuplePrefixComparator<k, k> {
00965     template <class Tuple1, class Tuple2>
00966     static bool Eq(const Tuple1& t1, const Tuple2& t2) {
00967         return SameSizeTuplePrefixComparator<k - 1, k - 1>::Eq(t1, t2) &&
00968             ::std::tr1::get<k - 1>(t1) == ::std::tr1::get<k - 1>(t2);
00969     }
00970 };
00971
00972 } // namespace gtest_internal
00973
00974 template <GTEST_10_TYPENAMES_(T), GTEST_10_TYPENAMES_(U)>
00975 inline bool operator==(const GTEST_10_TUPLE_(T)& t,
00976                           const GTEST_10_TUPLE_(U)& u) {
00977     return gtest_internal::SameSizeTuplePrefixComparator<
00978             tuple_size<GTEST_10_TUPLE_(T) >::value,
```

```

00979     tuple_size<GTEST_10_TUPLE_(U) >::value>::Eq(t, u);
00980 }
00981
00982 template <GTEST_10_TYPENAMES_(T), GTEST_10_TYPENAMES_(U)>
00983 inline bool operator!=(const GTEST_10_TUPLE_(T)& t,
00984                         const GTEST_10_TUPLE_(U)& u) { return !(t == u); }
00985
00986 // 6.1.4 Pairs.
00987 // Unimplemented.
00988
00989 } // namespace tr1
00990 } // namespace std
00991
00992 #undef GTEST_0_TUPLE_
00993 #undef GTEST_1_TUPLE_
00994 #undef GTEST_2_TUPLE_
00995 #undef GTEST_3_TUPLE_
00996 #undef GTEST_4_TUPLE_
00997 #undef GTEST_5_TUPLE_
00998 #undef GTEST_6_TUPLE_
00999 #undef GTEST_7_TUPLE_
01000 #undef GTEST_8_TUPLE_
01001 #undef GTEST_9_TUPLE_
01002 #undef GTEST_10_TUPLE_
01003
01004 #undef GTEST_0_TYPENAMES_
01005 #undef GTEST_1_TYPENAMES_
01006 #undef GTEST_2_TYPENAMES_
01007 #undef GTEST_3_TYPENAMES_
01008 #undef GTEST_4_TYPENAMES_
01009 #undef GTEST_5_TYPENAMES_
01010 #undef GTEST_6_TYPENAMES_
01011 #undef GTEST_7_TYPENAMES_
01012 #undef GTEST_8_TYPENAMES_
01013 #undef GTEST_9_TYPENAMES_
01014 #undef GTEST_10_TYPENAMES_
01015
01016 #undef GTEST_DECLARE_TUPLE_AS_FRIEND_
01017 #undef GTEST_BY_REF_
01018 #undef GTEST_ADD_REF_
01019 #undef GTEST_TUPLE_ELEMENT_
01020
01021 #endif // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_TUPLE_H_

```

9.51 Dokumentacja pliku

packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-type-util.h

```
#include "gtest/internal/gtest-port.h"
```

Przestrzenie nazw

- namespace **testing**
- namespace **testing::internal**

Funkcje

- std::string **testing::internal::CanonicalizeForStdLibVersioning** (std::string s)
- template<typename T>
std::string **testing::internal::GetTypeName** ()

9.52 gtest-type-util.h

Idź do dokumentacji tego pliku.

```
00001 // This file was GENERATED by command:  
00002 //     pump.py gtest-type-util.h.pump  
00003 // DO NOT EDIT BY HAND!!!  
00004  
00005 // Copyright 2008 Google Inc.  
00006 // All Rights Reserved.  
00007 //  
00008 // Redistribution and use in source and binary forms, with or without  
00009 // modification, are permitted provided that the following conditions are  
00010 // met:  
00011 //  
00012 //     * Redistributions of source code must retain the above copyright  
00013 // notice, this list of conditions and the following disclaimer.  
00014 //     * Redistributions in binary form must reproduce the above  
00015 // copyright notice, this list of conditions and the following disclaimer  
00016 // in the documentation and/or other materials provided with the  
00017 // distribution.  
00018 //     * Neither the name of Google Inc. nor the names of its  
00019 // contributors may be used to endorse or promote products derived from  
00020 // this software without specific prior written permission.  
00021 //  
00022 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS  
00023 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT  
00024 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR  
00025 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT  
00026 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,  
00027 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT  
00028 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,  
00029 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY  
00030 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
00031 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE  
00032 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
00033  
00034  
00035 // Type utilities needed for implementing typed and type-parameterized  
00036 // tests. This file is generated by a SCRIPT. DO NOT EDIT BY HAND!  
00037 //  
00038 // Currently we support at most 50 types in a list, and at most 50  
00039 // type-parameterized tests in one type-parameterized test case.  
00040 // Please contact googletestframework@googlegroups.com if you need  
00041 // more.  
00042  
00043 // GOOGLETEST_CM0001 DO NOT DELETE  
00044  
00045 #ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_TYPE_UTIL_H_  
00046 #define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_TYPE_UTIL_H_  
00047  
00048 #include "gtest/internal/gtest-port.h"  
00049  
00050 // #ifdef __GNUC__ is too general here. It is possible to use gcc without using  
00051 // libstdc++ (which is where cxxabi.h comes from).  
00052 #if GTEST_HAS_CXXABI_H_  
00053 # include <cxxabi.h>  
00054 # elif defined(__HP_aCC)  
00055 # include <acxx_demangle.h>  
00056 # endif // GTEST_HASH_CXXABI_H_  
00057  
00058 namespace testing {  
00059 namespace internal {  
00060  
00061 // Canonicalizes a given name with respect to the Standard C++ Library.  
00062 // This handles removing the inline namespace within `std` that is  
00063 // used by various standard libraries (e.g., `std::__1`). Names outside  
00064 // of namespace std are returned unmodified.  
00065 inline std::string CanonicalizeForStdLibVersioning(std::string s) {  
00066     static const char prefix[] = "std::__";  
00067     if (s.compare(0, strlen(prefix), prefix) == 0) {  
00068         std::string::size_type end = s.find(":::", strlen(prefix));  
00069         if (end != snpos) {  
00070             // Erase everything between the initial `std` and the second `::`.  
00071             s.erase(strlen("std"), end - strlen("std"));  
00072         }  
00073     }  
00074     return s;  
00075 }  
00076  
00077 // GetTypeName<T>() returns a human-readable name of type T.  
00078 // NB: This function is also used in Google Mock, so don't move it inside of  
00079 // the typed-test-only section below.  
00080 template <typename T>  
00081 std::string GetTypeName() {  
00082 # if GTEST_HAS_RTTI
```

```
00083
00084     const char* const name = typeid(T).name();
00085 # if GTEST_HAS_CXXABI_H_ || defined(__HP_aCC)
00086     int status = 0;
00087 // gcc's implementation of typeid(T).name() mangles the type name,
00088 // so we have to demangle it.
00089 # if GTEST_HAS_CXXABI_H_
00090     using abi::__cxa_demangle;
00091 # endif // GTEST_HAS_CXXABI_H_
00092     char* const readable_name = __cxa_demangle(name, 0, 0, &status);
00093     const std::string name_str(status == 0 ? readable_name : name);
00094     free(readable_name);
00095     return CanonicalizeForStdLibVersioning(name_str);
00096 # else
00097     return name;
00098 # endif // GTEST_HAS_CXXABI_H_ || __HP_aCC
00099
00100 # else
00101
00102     return "<type>";
00103
00104 # endif // GTEST_HAS_RTTI
00105 }
00106
00107 #if GTEST_HAS_TYPED_TEST || GTEST_HAS_TYPED_TEST_P
00108
00109 // AssertTypeEq<T1, T2>::type is defined iff T1 and T2 are the same
00110 // type. This can be used as a compile-time assertion to ensure that
00111 // two types are equal.
00112
00113 template <typename T1, typename T2>
00114 struct AssertTypeEq;
00115
00116 template <typename T>
00117 struct AssertTypeEq<T, T> {
00118     typedef bool type;
00119 };
00120
00121 // A unique type used as the default value for the arguments of class
00122 // template Types. This allows us to simulate variadic templates
00123 // (e.g. Types<int>, Type<int, double>, and etc), which C++ doesn't
00124 // support directly.
00125 struct None {};
00126
00127 // The following family of struct and struct templates are used to
00128 // represent type lists. In particular, TypesN<T1, T2, ..., TN>
00129 // represents a type list with N types (T1, T2, ..., and TN) in it.
00130 // Except for Types0, every struct in the family has two member types:
00131 // Head for the first type in the list, and Tail for the rest of the
00132 // list.
00133
00134 // The empty type list.
00135 struct Types0 {};
00136
00137 // Type lists of length 1, 2, 3, and so on.
00138
00139 template <typename T1>
00140 struct Types1 {
00141     typedef T1 Head;
00142     typedef Types0 Tail;
00143 };
00144 template <typename T1, typename T2>
00145 struct Types2 {
00146     typedef T1 Head;
00147     typedef Types1<T2> Tail;
00148 };
00149
00150 template <typename T1, typename T2, typename T3>
00151 struct Types3 {
00152     typedef T1 Head;
00153     typedef Types2<T2, T3> Tail;
00154 };
00155
00156 template <typename T1, typename T2, typename T3, typename T4>
00157 struct Types4 {
00158     typedef T1 Head;
00159     typedef Types3<T2, T3, T4> Tail;
00160 };
00161
00162 template <typename T1, typename T2, typename T3, typename T4, typename T5>
00163 struct Types5 {
00164     typedef T1 Head;
00165     typedef Types4<T2, T3, T4, T5> Tail;
00166 };
00167
00168 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00169           typename T6>
```

```
00170 struct Types6 {
00171     typedef T1 Head;
00172     typedef Types5<T2, T3, T4, T5, T6> Tail;
00173 };
00174
00175 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00176             typename T6, typename T7>
00177 struct Types7 {
00178     typedef T1 Head;
00179     typedef Types6<T2, T3, T4, T5, T6, T7> Tail;
00180 };
00181
00182 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00183             typename T6, typename T7, typename T8>
00184 struct Types8 {
00185     typedef T1 Head;
00186     typedef Types7<T2, T3, T4, T5, T6, T7, T8> Tail;
00187 };
00188
00189 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00190             typename T6, typename T7, typename T8, typename T9>
00191 struct Types9 {
00192     typedef T1 Head;
00193     typedef Types8<T2, T3, T4, T5, T6, T7, T8, T9> Tail;
00194 };
00195
00196 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00197             typename T6, typename T7, typename T8, typename T9, typename T10>
00198 struct Types10 {
00199     typedef T1 Head;
00200     typedef Types9<T2, T3, T4, T5, T6, T7, T8, T9, T10> Tail;
00201 };
00202
00203 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00204             typename T6, typename T7, typename T8, typename T9, typename T10,
00205             typename T11>
00206 struct Types11 {
00207     typedef T1 Head;
00208     typedef Types10<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11> Tail;
00209 };
00210
00211 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00212             typename T6, typename T7, typename T8, typename T9, typename T10,
00213             typename T11, typename T12>
00214 struct Types12 {
00215     typedef T1 Head;
00216     typedef Types11<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12> Tail;
00217 };
00218
00219 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00220             typename T6, typename T7, typename T8, typename T9, typename T10,
00221             typename T11, typename T12, typename T13>
00222 struct Types13 {
00223     typedef T1 Head;
00224     typedef Types12<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13> Tail;
00225 };
00226
00227 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00228             typename T6, typename T7, typename T8, typename T9, typename T10,
00229             typename T11, typename T12, typename T13, typename T14>
00230 struct Types14 {
00231     typedef T1 Head;
00232     typedef Types13<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14> Tail;
00233 };
00234
00235 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00236             typename T6, typename T7, typename T8, typename T9, typename T10,
00237             typename T11, typename T12, typename T13, typename T14, typename T15>
00238 struct Types15 {
00239     typedef T1 Head;
00240     typedef Types14<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
00241                 T15> Tail;
00242 };
00243
00244 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00245             typename T6, typename T7, typename T8, typename T9, typename T10,
00246             typename T11, typename T12, typename T13, typename T14, typename T15,
00247             typename T16>
00248 struct Types16 {
00249     typedef T1 Head;
00250     typedef Types15<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00251                 T16> Tail;
00252 };
00253
00254 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00255             typename T6, typename T7, typename T8, typename T9, typename T10,
00256             typename T11, typename T12, typename T13, typename T14, typename T15,
```

```
00257     typename T16, typename T17>
00258 struct Types17 {
00259     typedef T1 Head;
00260     typedef Types16<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00261             T16, T17> Tail;
00262 };
00263
00264 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00265         typename T6, typename T7, typename T8, typename T9, typename T10,
00266         typename T11, typename T12, typename T13, typename T14, typename T15,
00267         typename T16, typename T17, typename T18>
00268 struct Types18 {
00269     typedef T1 Head;
00270     typedef Types17<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00271             T16, T17, T18> Tail;
00272 };
00273
00274 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00275         typename T6, typename T7, typename T8, typename T9, typename T10,
00276         typename T11, typename T12, typename T13, typename T14, typename T15,
00277         typename T16, typename T17, typename T18, typename T19>
00278 struct Types19 {
00279     typedef T1 Head;
00280     typedef Types18<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00281             T16, T17, T18, T19> Tail;
00282 };
00283
00284 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00285         typename T6, typename T7, typename T8, typename T9, typename T10,
00286         typename T11, typename T12, typename T13, typename T14, typename T15,
00287         typename T16, typename T17, typename T18, typename T19, typename T20>
00288 struct Types20 {
00289     typedef T1 Head;
00290     typedef Types19<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00291             T16, T17, T18, T19, T20> Tail;
00292 };
00293
00294 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00295         typename T6, typename T7, typename T8, typename T9, typename T10,
00296         typename T11, typename T12, typename T13, typename T14, typename T15,
00297         typename T16, typename T17, typename T18, typename T19, typename T20,
00298         typename T21>
00299 struct Types21 {
00300     typedef T1 Head;
00301     typedef Types20<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00302             T16, T17, T18, T19, T20, T21> Tail;
00303 };
00304
00305 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00306         typename T6, typename T7, typename T8, typename T9, typename T10,
00307         typename T11, typename T12, typename T13, typename T14, typename T15,
00308         typename T16, typename T17, typename T18, typename T19, typename T20,
00309         typename T21, typename T22>
00310 struct Types22 {
00311     typedef T1 Head;
00312     typedef Types21<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00313             T16, T17, T18, T19, T20, T21, T22> Tail;
00314 };
00315
00316 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00317         typename T6, typename T7, typename T8, typename T9, typename T10,
00318         typename T11, typename T12, typename T13, typename T14, typename T15,
00319         typename T16, typename T17, typename T18, typename T19, typename T20,
00320         typename T21, typename T22, typename T23>
00321 struct Types23 {
00322     typedef T1 Head;
00323     typedef Types22<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00324             T16, T17, T18, T19, T20, T21, T22, T23> Tail;
00325 };
00326
00327 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00328         typename T6, typename T7, typename T8, typename T9, typename T10,
00329         typename T11, typename T12, typename T13, typename T14, typename T15,
00330         typename T16, typename T17, typename T18, typename T19, typename T20,
00331         typename T21, typename T22, typename T23, typename T24>
00332 struct Types24 {
00333     typedef T1 Head;
00334     typedef Types23<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00335             T16, T17, T18, T19, T20, T21, T22, T23, T24> Tail;
00336 };
00337
00338 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00339         typename T6, typename T7, typename T8, typename T9, typename T10,
00340         typename T11, typename T12, typename T13, typename T14, typename T15,
00341         typename T16, typename T17, typename T18, typename T19, typename T20,
00342         typename T21, typename T22, typename T23, typename T24, typename T25>
00343 struct Types25 {
```

```
00344     typedef T1 Head;
00345     typedef Types24<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00346         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25> Tail;
00347 };
00348
00349 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00350     typename T6, typename T7, typename T8, typename T9, typename T10,
00351     typename T11, typename T12, typename T13, typename T14, typename T15,
00352     typename T16, typename T17, typename T18, typename T19, typename T20,
00353     typename T21, typename T22, typename T23, typename T24, typename T25,
00354     typename T26>
00355     struct Types26 {
00356         typedef T1 Head;
00357         typedef Types25<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00358             T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26> Tail;
00359 };
00360
00361 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00362     typename T6, typename T7, typename T8, typename T9, typename T10,
00363     typename T11, typename T12, typename T13, typename T14, typename T15,
00364     typename T16, typename T17, typename T18, typename T19, typename T20,
00365     typename T21, typename T22, typename T23, typename T24, typename T25,
00366     typename T26, typename T27>
00367     struct Types27 {
00368         typedef T1 Head;
00369         typedef Types26<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00370             T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27> Tail;
00371 };
00372
00373 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00374     typename T6, typename T7, typename T8, typename T9, typename T10,
00375     typename T11, typename T12, typename T13, typename T14, typename T15,
00376     typename T16, typename T17, typename T18, typename T19, typename T20,
00377     typename T21, typename T22, typename T23, typename T24, typename T25,
00378     typename T26, typename T27, typename T28>
00379     struct Types28 {
00380         typedef T1 Head;
00381         typedef Types27<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00382             T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28> Tail;
00383 };
00384
00385 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00386     typename T6, typename T7, typename T8, typename T9, typename T10,
00387     typename T11, typename T12, typename T13, typename T14, typename T15,
00388     typename T16, typename T17, typename T18, typename T19, typename T20,
00389     typename T21, typename T22, typename T23, typename T24, typename T25,
00390     typename T26, typename T27, typename T28, typename T29>
00391     struct Types29 {
00392         typedef T1 Head;
00393         typedef Types28<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00394             T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
00395             T29> Tail;
00396 };
00397
00398 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00399     typename T6, typename T7, typename T8, typename T9, typename T10,
00400     typename T11, typename T12, typename T13, typename T14, typename T15,
00401     typename T16, typename T17, typename T18, typename T19, typename T20,
00402     typename T21, typename T22, typename T23, typename T24, typename T25,
00403     typename T26, typename T27, typename T28, typename T29, typename T30>
00404     struct Types30 {
00405         typedef T1 Head;
00406         typedef Types29<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00407             T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00408             T30> Tail;
00409 };
00410
00411 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00412     typename T6, typename T7, typename T8, typename T9, typename T10,
00413     typename T11, typename T12, typename T13, typename T14, typename T15,
00414     typename T16, typename T17, typename T18, typename T19, typename T20,
00415     typename T21, typename T22, typename T23, typename T24, typename T25,
00416     typename T26, typename T27, typename T28, typename T29, typename T30,
00417     typename T31>
00418     struct Types31 {
00419         typedef T1 Head;
00420         typedef Types30<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00421             T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00422             T30, T31> Tail;
00423 };
00424
00425 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00426     typename T6, typename T7, typename T8, typename T9, typename T10,
00427     typename T11, typename T12, typename T13, typename T14, typename T15,
00428     typename T16, typename T17, typename T18, typename T19, typename T20,
00429     typename T21, typename T22, typename T23, typename T24, typename T25,
00430     typename T26, typename T27, typename T28, typename T29, typename T30,
```

```

00431     typename T31, typename T32>
00432 struct Types32 {
00433     typedef T1 Head;
00434     typedef Types31<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00435         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00436         T30, T31, T32> Tail;
00437 };
00438
00439 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00440     typename T6, typename T7, typename T8, typename T9, typename T10,
00441     typename T11, typename T12, typename T13, typename T14, typename T15,
00442     typename T16, typename T17, typename T18, typename T19, typename T20,
00443     typename T21, typename T22, typename T23, typename T24, typename T25,
00444     typename T26, typename T27, typename T28, typename T29, typename T30,
00445     typename T31, typename T32, typename T33>
00446 struct Types33 {
00447     typedef T1 Head;
00448     typedef Types32<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00449         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00450         T30, T31, T32, T33> Tail;
00451 };
00452
00453 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00454     typename T6, typename T7, typename T8, typename T9, typename T10,
00455     typename T11, typename T12, typename T13, typename T14, typename T15,
00456     typename T16, typename T17, typename T18, typename T19, typename T20,
00457     typename T21, typename T22, typename T23, typename T24, typename T25,
00458     typename T26, typename T27, typename T28, typename T29, typename T30,
00459     typename T31, typename T32, typename T33, typename T34>
00460 struct Types34 {
00461     typedef T1 Head;
00462     typedef Types33<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00463         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00464         T30, T31, T32, T33, T34> Tail;
00465 };
00466
00467 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00468     typename T6, typename T7, typename T8, typename T9, typename T10,
00469     typename T11, typename T12, typename T13, typename T14, typename T15,
00470     typename T16, typename T17, typename T18, typename T19, typename T20,
00471     typename T21, typename T22, typename T23, typename T24, typename T25,
00472     typename T26, typename T27, typename T28, typename T29, typename T30,
00473     typename T31, typename T32, typename T33, typename T34, typename T35>
00474 struct Types35 {
00475     typedef T1 Head;
00476     typedef Types34<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00477         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00478         T30, T31, T32, T33, T34, T35> Tail;
00479 };
00480
00481 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00482     typename T6, typename T7, typename T8, typename T9, typename T10,
00483     typename T11, typename T12, typename T13, typename T14, typename T15,
00484     typename T16, typename T17, typename T18, typename T19, typename T20,
00485     typename T21, typename T22, typename T23, typename T24, typename T25,
00486     typename T26, typename T27, typename T28, typename T29, typename T30,
00487     typename T31, typename T32, typename T33, typename T34, typename T35,
00488     typename T36>
00489 struct Types36 {
00490     typedef T1 Head;
00491     typedef Types35<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00492         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00493         T30, T31, T32, T33, T34, T35, T36> Tail;
00494 };
00495
00496 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00497     typename T6, typename T7, typename T8, typename T9, typename T10,
00498     typename T11, typename T12, typename T13, typename T14, typename T15,
00499     typename T16, typename T17, typename T18, typename T19, typename T20,
00500     typename T21, typename T22, typename T23, typename T24, typename T25,
00501     typename T26, typename T27, typename T28, typename T29, typename T30,
00502     typename T31, typename T32, typename T33, typename T34, typename T35,
00503     typename T36, typename T37>
00504 struct Types37 {
00505     typedef T1 Head;
00506     typedef Types36<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00507         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00508         T30, T31, T32, T33, T34, T35, T36, T37> Tail;
00509 };
00510
00511 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00512     typename T6, typename T7, typename T8, typename T9, typename T10,
00513     typename T11, typename T12, typename T13, typename T14, typename T15,
00514     typename T16, typename T17, typename T18, typename T19, typename T20,
00515     typename T21, typename T22, typename T23, typename T24, typename T25,
00516     typename T26, typename T27, typename T28, typename T29, typename T30,
00517     typename T31, typename T32, typename T33, typename T34, typename T35,

```

```
00518     typename T36, typename T37, typename T38>
00519 struct Types38 {
00520     typedef T1 Head;
00521     typedef Types37<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00522         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00523         T30, T31, T32, T33, T34, T35, T36, T37, T38> Tail;
00524 };
00525
00526 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00527     typename T6, typename T7, typename T8, typename T9, typename T10,
00528     typename T11, typename T12, typename T13, typename T14, typename T15,
00529     typename T16, typename T17, typename T18, typename T19, typename T20,
00530     typename T21, typename T22, typename T23, typename T24, typename T25,
00531     typename T26, typename T27, typename T28, typename T29, typename T30,
00532     typename T31, typename T32, typename T33, typename T34, typename T35,
00533     typename T36, typename T37, typename T38, typename T39>
00534 struct Types39 {
00535     typedef T1 Head;
00536     typedef Types38<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00537         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00538         T30, T31, T32, T33, T34, T35, T36, T37, T38, T39> Tail;
00539 };
00540
00541 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00542     typename T6, typename T7, typename T8, typename T9, typename T10,
00543     typename T11, typename T12, typename T13, typename T14, typename T15,
00544     typename T16, typename T17, typename T18, typename T19, typename T20,
00545     typename T21, typename T22, typename T23, typename T24, typename T25,
00546     typename T26, typename T27, typename T28, typename T29, typename T30,
00547     typename T31, typename T32, typename T33, typename T34, typename T35,
00548     typename T36, typename T37, typename T38, typename T39, typename T40>
00549 struct Types40 {
00550     typedef T1 Head;
00551     typedef Types39<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00552         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00553         T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40> Tail;
00554 };
00555
00556 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00557     typename T6, typename T7, typename T8, typename T9, typename T10,
00558     typename T11, typename T12, typename T13, typename T14, typename T15,
00559     typename T16, typename T17, typename T18, typename T19, typename T20,
00560     typename T21, typename T22, typename T23, typename T24, typename T25,
00561     typename T26, typename T27, typename T28, typename T29, typename T30,
00562     typename T31, typename T32, typename T33, typename T34, typename T35,
00563     typename T36, typename T37, typename T38, typename T39, typename T40,
00564     typename T41>
00565 struct Types41 {
00566     typedef T1 Head;
00567     typedef Types40<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00568         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00569         T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41> Tail;
00570 };
00571
00572 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00573     typename T6, typename T7, typename T8, typename T9, typename T10,
00574     typename T11, typename T12, typename T13, typename T14, typename T15,
00575     typename T16, typename T17, typename T18, typename T19, typename T20,
00576     typename T21, typename T22, typename T23, typename T24, typename T25,
00577     typename T26, typename T27, typename T28, typename T29, typename T30,
00578     typename T31, typename T32, typename T33, typename T34, typename T35,
00579     typename T36, typename T37, typename T38, typename T39, typename T40,
00580     typename T41, typename T42>
00581 struct Types42 {
00582     typedef T1 Head;
00583     typedef Types41<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00584         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00585         T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42> Tail;
00586 };
00587
00588 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00589     typename T6, typename T7, typename T8, typename T9, typename T10,
00590     typename T11, typename T12, typename T13, typename T14, typename T15,
00591     typename T16, typename T17, typename T18, typename T19, typename T20,
00592     typename T21, typename T22, typename T23, typename T24, typename T25,
00593     typename T26, typename T27, typename T28, typename T29, typename T30,
00594     typename T31, typename T32, typename T33, typename T34, typename T35,
00595     typename T36, typename T37, typename T38, typename T39, typename T40,
00596     typename T41, typename T42, typename T43>
00597 struct Types43 {
00598     typedef T1 Head;
00599     typedef Types42<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00600         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00601         T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42,
00602         T43> Tail;
00603 };
00604
```

```
00605 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00606     typename T6, typename T7, typename T8, typename T9, typename T10,
00607     typename T11, typename T12, typename T13, typename T14, typename T15,
00608     typename T16, typename T17, typename T18, typename T19, typename T20,
00609     typename T21, typename T22, typename T23, typename T24, typename T25,
00610     typename T26, typename T27, typename T28, typename T29, typename T30,
00611     typename T31, typename T32, typename T33, typename T34, typename T35,
00612     typename T36, typename T37, typename T38, typename T39, typename T40,
00613     typename T41, typename T42, typename T43, typename T44>
00614 struct Types44 {
00615     typedef T1 Head;
00616     typedef Types43<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00617         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00618         T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
00619         T44> Tail;
00620 };
00621
00622 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00623     typename T6, typename T7, typename T8, typename T9, typename T10,
00624     typename T11, typename T12, typename T13, typename T14, typename T15,
00625     typename T16, typename T17, typename T18, typename T19, typename T20,
00626     typename T21, typename T22, typename T23, typename T24, typename T25,
00627     typename T26, typename T27, typename T28, typename T29, typename T30,
00628     typename T31, typename T32, typename T33, typename T34, typename T35,
00629     typename T36, typename T37, typename T38, typename T39, typename T40,
00630     typename T41, typename T42, typename T43, typename T44, typename T45>
00631 struct Types45 {
00632     typedef T1 Head;
00633     typedef Types44<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00634         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00635         T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
00636         T44, T45> Tail;
00637 };
00638
00639 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00640     typename T6, typename T7, typename T8, typename T9, typename T10,
00641     typename T11, typename T12, typename T13, typename T14, typename T15,
00642     typename T16, typename T17, typename T18, typename T19, typename T20,
00643     typename T21, typename T22, typename T23, typename T24, typename T25,
00644     typename T26, typename T27, typename T28, typename T29, typename T30,
00645     typename T31, typename T32, typename T33, typename T34, typename T35,
00646     typename T36, typename T37, typename T38, typename T39, typename T40,
00647     typename T41, typename T42, typename T43, typename T44, typename T45,
00648     typename T46>
00649 struct Types46 {
00650     typedef T1 Head;
00651     typedef Types45<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00652         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00653         T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
00654         T44, T45, T46> Tail;
00655 };
00656
00657 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00658     typename T6, typename T7, typename T8, typename T9, typename T10,
00659     typename T11, typename T12, typename T13, typename T14, typename T15,
00660     typename T16, typename T17, typename T18, typename T19, typename T20,
00661     typename T21, typename T22, typename T23, typename T24, typename T25,
00662     typename T26, typename T27, typename T28, typename T29, typename T30,
00663     typename T31, typename T32, typename T33, typename T34, typename T35,
00664     typename T36, typename T37, typename T38, typename T39, typename T40,
00665     typename T41, typename T42, typename T43, typename T44, typename T45,
00666     typename T46, typename T47>
00667 struct Types47 {
00668     typedef T1 Head;
00669     typedef Types46<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00670         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00671         T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
00672         T44, T45, T46, T47> Tail;
00673 };
00674
00675 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00676     typename T6, typename T7, typename T8, typename T9, typename T10,
00677     typename T11, typename T12, typename T13, typename T14, typename T15,
00678     typename T16, typename T17, typename T18, typename T19, typename T20,
00679     typename T21, typename T22, typename T23, typename T24, typename T25,
00680     typename T26, typename T27, typename T28, typename T29, typename T30,
00681     typename T31, typename T32, typename T33, typename T34, typename T35,
00682     typename T36, typename T37, typename T38, typename T39, typename T40,
00683     typename T41, typename T42, typename T43, typename T44, typename T45,
00684     typename T46, typename T47, typename T48>
00685 struct Types48 {
00686     typedef T1 Head;
00687     typedef Types47<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00688         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00689         T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
00690         T44, T45, T46, T47, T48> Tail;
00691 };
```

```

00692
00693 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00694     typename T6, typename T7, typename T8, typename T9, typename T10,
00695     typename T11, typename T12, typename T13, typename T14, typename T15,
00696     typename T16, typename T17, typename T18, typename T19, typename T20,
00697     typename T21, typename T22, typename T23, typename T24, typename T25,
00698     typename T26, typename T27, typename T28, typename T29, typename T30,
00699     typename T31, typename T32, typename T33, typename T34, typename T35,
00700     typename T36, typename T37, typename T38, typename T39, typename T40,
00701     typename T41, typename T42, typename T43, typename T44, typename T45,
00702     typename T46, typename T47, typename T48, typename T49>
00703 struct Types49 {
00704     typedef T1 Head;
00705     typedef Types48<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00706         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00707         T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
00708         T44, T45, T46, T47, T48, T49> Tail;
00709 };
00710
00711 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00712     typename T6, typename T7, typename T8, typename T9, typename T10,
00713     typename T11, typename T12, typename T13, typename T14, typename T15,
00714     typename T16, typename T17, typename T18, typename T19, typename T20,
00715     typename T21, typename T22, typename T23, typename T24, typename T25,
00716     typename T26, typename T27, typename T28, typename T29, typename T30,
00717     typename T31, typename T32, typename T33, typename T34, typename T35,
00718     typename T36, typename T37, typename T38, typename T39, typename T40,
00719     typename T41, typename T42, typename T43, typename T44, typename T45,
00720     typename T46, typename T47, typename T48, typename T49, typename T50>
00721 struct Types50 {
00722     typedef T1 Head;
00723     typedef Types49<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
00724         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
00725         T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
00726         T44, T45, T46, T47, T48, T49, T50> Tail;
00727 };
00728
00729
00730 } // namespace internal
00731
00732 // We don't want to require the users to write TypesN<...> directly,
00733 // as that would require them to count the length. Types<...> is much
00734 // easier to write, but generates horrible messages when there is a
00735 // compiler error, as gcc insists on printing out each template
00736 // argument, even if it has the default value (this means Types<int>
00737 // will appear as Types<int, None, None, ..., None> in the compiler
00738 // errors).
00739 //
00740 // Our solution is to combine the best part of the two approaches: a
00741 // user would write Types<T1, ..., TN>, and Google Test will translate
00742 // that to TypesN<T1, ..., TN> internally to make error messages
00743 // readable. The translation is done by the 'type' member of the
00744 // Types template.
00745 template <typename T1 = internal::None, typename T2 = internal::None,
00746     typename T3 = internal::None, typename T4 = internal::None,
00747     typename T5 = internal::None, typename T6 = internal::None,
00748     typename T7 = internal::None, typename T8 = internal::None,
00749     typename T9 = internal::None, typename T10 = internal::None,
00750     typename T11 = internal::None, typename T12 = internal::None,
00751     typename T13 = internal::None, typename T14 = internal::None,
00752     typename T15 = internal::None, typename T16 = internal::None,
00753     typename T17 = internal::None, typename T18 = internal::None,
00754     typename T19 = internal::None, typename T20 = internal::None,
00755     typename T21 = internal::None, typename T22 = internal::None,
00756     typename T23 = internal::None, typename T24 = internal::None,
00757     typename T25 = internal::None, typename T26 = internal::None,
00758     typename T27 = internal::None, typename T28 = internal::None,
00759     typename T29 = internal::None, typename T30 = internal::None,
00760     typename T31 = internal::None, typename T32 = internal::None,
00761     typename T33 = internal::None, typename T34 = internal::None,
00762     typename T35 = internal::None, typename T36 = internal::None,
00763     typename T37 = internal::None, typename T38 = internal::None,
00764     typename T39 = internal::None, typename T40 = internal::None,
00765     typename T41 = internal::None, typename T42 = internal::None,
00766     typename T43 = internal::None, typename T44 = internal::None,
00767     typename T45 = internal::None, typename T46 = internal::None,
00768     typename T47 = internal::None, typename T48 = internal::None,
00769     typename T49 = internal::None, typename T50 = internal::None>
00770 struct Types {
00771     typedef internal::Types50<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
00772         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
00773         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40,
00774         T41, T42, T43, T44, T45, T46, T47, T48, T49, T50> type;
00775 };
00776
00777 template <>
00778 struct Types<internal::None, internal::None, internal::None, internal::None,

```



```

00953     internal::None, internal::None, internal::None, internal::None,
00954     internal::None, internal::None, internal::None, internal::None,
00955     internal::None, internal::None, internal::None, internal::None,
00956     internal::None, internal::None, internal::None, internal::None,
00957     internal::None, internal::None, internal::None, internal::None,
00958     internal::None, internal::None, internal::None, internal::None,
00959     internal::None, internal::None, internal::None, internal::None,
00960     internal::None, internal::None> {
00961     typedef internal::Types11<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11> type;
00962 };
00963 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00964     typename T6, typename T7, typename T8, typename T9, typename T10,
00965     typename T11, typename T12>
00966     struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, internal::None,
00967     internal::None, internal::None, internal::None, internal::None,
00968     internal::None, internal::None, internal::None, internal::None,
00969     internal::None, internal::None, internal::None, internal::None,
00970     internal::None, internal::None, internal::None, internal::None,
00971     internal::None, internal::None, internal::None, internal::None,
00972     internal::None, internal::None, internal::None, internal::None,
00973     internal::None, internal::None, internal::None, internal::None,
00974     internal::None, internal::None, internal::None, internal::None,
00975     internal::None, internal::None, internal::None, internal::None,
00976     internal::None> {
00977     typedef internal::Types12<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
00978     T12> type;
00979 };
00980 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00981     typename T6, typename T7, typename T8, typename T9, typename T10,
00982     typename T11, typename T12, typename T13>
00983     struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
00984     internal::None, internal::None, internal::None, internal::None,
00985     internal::None, internal::None, internal::None, internal::None,
00986     internal::None, internal::None, internal::None, internal::None,
00987     internal::None, internal::None, internal::None, internal::None,
00988     internal::None, internal::None, internal::None, internal::None,
00989     internal::None, internal::None, internal::None, internal::None,
00990     internal::None, internal::None, internal::None, internal::None,
00991     internal::None, internal::None, internal::None, internal::None,
00992     internal::None, internal::None, internal::None, internal::None,
00993     internal::None> {
00994     typedef internal::Types13<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
00995     T13> type;
00996 };
00997 template <typename T1, typename T2, typename T3, typename T4, typename T5,
00998     typename T6, typename T7, typename T8, typename T9, typename T10,
00999     typename T11, typename T12, typename T13, typename T14>
01000     struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01001     internal::None, internal::None, internal::None, internal::None,
01002     internal::None, internal::None, internal::None, internal::None,
01003     internal::None, internal::None, internal::None, internal::None,
01004     internal::None, internal::None, internal::None, internal::None,
01005     internal::None, internal::None, internal::None, internal::None,
01006     internal::None, internal::None, internal::None, internal::None,
01007     internal::None, internal::None, internal::None, internal::None,
01008     internal::None, internal::None, internal::None, internal::None,
01009     internal::None, internal::None, internal::None, internal::None> {
01010     typedef internal::Types14<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01011     T13, T14> type;
01012 };
01013 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01014     typename T6, typename T7, typename T8, typename T9, typename T10,
01015     typename T11, typename T12, typename T13, typename T14, typename T15>
01016     struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01017     internal::None, internal::None, internal::None, internal::None,
01018     internal::None, internal::None, internal::None, internal::None,
01019     internal::None, internal::None, internal::None, internal::None,
01020     internal::None, internal::None, internal::None, internal::None,
01021     internal::None, internal::None, internal::None, internal::None,
01022     internal::None, internal::None, internal::None, internal::None,
01023     internal::None, internal::None, internal::None, internal::None,
01024     internal::None, internal::None, internal::None, internal::None,
01025     internal::None, internal::None, internal::None> {
01026     typedef internal::Types15<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01027     T13, T14, T15> type;
01028 };
01029 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01030     typename T6, typename T7, typename T8, typename T9, typename T10,
01031     typename T11, typename T12, typename T13, typename T14, typename T15,
01032     typename T16>
01033     struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01034     T16, internal::None, internal::None, internal::None, internal::None,
01035     internal::None, internal::None, internal::None, internal::None,
01036     internal::None, internal::None, internal::None, internal::None,
01037     internal::None, internal::None, internal::None, internal::None,
01038     internal::None, internal::None, internal::None, internal::None,
01039     internal::None, internal::None, internal::None> {

```

```
01040     internal::None, internal::None, internal::None, internal::None,
01041     internal::None, internal::None, internal::None, internal::None,
01042     internal::None, internal::None> {
01043     typedef internal::Types16<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01044         T13, T14, T15, T16> type;
01045 };
01046 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01047     typename T6, typename T7, typename T8, typename T9, typename T10,
01048     typename T11, typename T12, typename T13, typename T14, typename T15,
01049     typename T16, typename T17>
01050 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01051     T16, T17, internal::None, internal::None, internal::None, internal::None,
01052     internal::None, internal::None, internal::None, internal::None,
01053     internal::None, internal::None, internal::None, internal::None,
01054     internal::None, internal::None, internal::None, internal::None,
01055     internal::None, internal::None, internal::None, internal::None,
01056     internal::None, internal::None, internal::None, internal::None,
01057     internal::None, internal::None, internal::None, internal::None,
01058     internal::None, internal::None, internal::None, internal::None,
01059     internal::None> {
01060     typedef internal::Types17<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01061         T13, T14, T15, T16, T17> type;
01062 };
01063 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01064     typename T6, typename T7, typename T8, typename T9, typename T10,
01065     typename T11, typename T12, typename T13, typename T14, typename T15,
01066     typename T16, typename T17, typename T18>
01067 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01068     T16, T17, T18, internal::None, internal::None, internal::None,
01069     internal::None, internal::None, internal::None, internal::None,
01070     internal::None, internal::None, internal::None, internal::None,
01071     internal::None, internal::None, internal::None, internal::None,
01072     internal::None, internal::None, internal::None, internal::None,
01073     internal::None, internal::None, internal::None, internal::None,
01074     internal::None, internal::None, internal::None, internal::None,
01075     internal::None, internal::None, internal::None, internal::None,
01076     internal::None> {
01077     typedef internal::Types18<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01078         T13, T14, T15, T16, T17, T18> type;
01079 };
01080 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01081     typename T6, typename T7, typename T8, typename T9, typename T10,
01082     typename T11, typename T12, typename T13, typename T14, typename T15,
01083     typename T16, typename T17, typename T18, typename T19>
01084 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01085     T16, T17, T18, T19, internal::None, internal::None, internal::None,
01086     internal::None, internal::None, internal::None, internal::None,
01087     internal::None, internal::None, internal::None, internal::None,
01088     internal::None, internal::None, internal::None, internal::None,
01089     internal::None, internal::None, internal::None, internal::None,
01090     internal::None, internal::None, internal::None, internal::None,
01091     internal::None, internal::None, internal::None, internal::None,
01092     internal::None, internal::None, internal::None, internal::None> {
01093     typedef internal::Types19<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01094         T13, T14, T15, T16, T17, T18, T19> type;
01095 };
01096 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01097     typename T6, typename T7, typename T8, typename T9, typename T10,
01098     typename T11, typename T12, typename T13, typename T14, typename T15,
01099     typename T16, typename T17, typename T18, typename T19, typename T20>
01100 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01101     T16, T17, T18, T19, T20, internal::None, internal::None, internal::None,
01102     internal::None, internal::None, internal::None, internal::None,
01103     internal::None, internal::None, internal::None, internal::None,
01104     internal::None, internal::None, internal::None, internal::None,
01105     internal::None, internal::None, internal::None, internal::None,
01106     internal::None, internal::None, internal::None, internal::None,
01107     internal::None, internal::None, internal::None, internal::None,
01108     internal::None, internal::None, internal::None> {
01109     typedef internal::Types20<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01110         T13, T14, T15, T16, T17, T18, T19, T20> type;
01111 };
01112 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01113     typename T6, typename T7, typename T8, typename T9, typename T10,
01114     typename T11, typename T12, typename T13, typename T14, typename T15,
01115     typename T16, typename T17, typename T18, typename T19, typename T20,
01116     typename T21>
01117 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01118     T16, T17, T18, T19, T20, T21, internal::None, internal::None,
01119     internal::None, internal::None, internal::None, internal::None,
01120     internal::None, internal::None, internal::None, internal::None,
01121     internal::None, internal::None, internal::None, internal::None,
01122     internal::None, internal::None, internal::None, internal::None,
01123     internal::None, internal::None, internal::None, internal::None,
01124     internal::None, internal::None, internal::None, internal::None,
01125     internal::None, internal::None, internal::None> {
01126     typedef internal::Types21<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
```

```

01127     T13, T14, T15, T16, T17, T18, T19, T20, T21> type;
01128 };
01129 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01130     typename T6, typename T7, typename T8, typename T9, typename T10,
01131     typename T11, typename T12, typename T13, typename T14, typename T15,
01132     typename T16, typename T17, typename T18, typename T19, typename T20,
01133     typename T21, typename T22>
01134 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01135     T16, T17, T18, T19, T20, T21, T22, internal::None, internal::None,
01136     internal::None, internal::None, internal::None, internal::None,
01137     internal::None, internal::None, internal::None, internal::None,
01138     internal::None, internal::None, internal::None, internal::None,
01139     internal::None, internal::None, internal::None, internal::None,
01140     internal::None, internal::None, internal::None, internal::None,
01141     internal::None, internal::None, internal::None, internal::None,
01142     internal::None, internal::None> {
01143     typedef internal::Types22<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01144         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22> type;
01145 };
01146 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01147     typename T6, typename T7, typename T8, typename T9, typename T10,
01148     typename T11, typename T12, typename T13, typename T14, typename T15,
01149     typename T16, typename T17, typename T18, typename T19, typename T20,
01150     typename T21, typename T22, typename T23>
01151 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01152     T16, T17, T18, T19, T20, T21, T22, T23, internal::None, internal::None,
01153     internal::None, internal::None, internal::None, internal::None,
01154     internal::None, internal::None, internal::None, internal::None,
01155     internal::None, internal::None, internal::None, internal::None,
01156     internal::None, internal::None, internal::None, internal::None,
01157     internal::None, internal::None, internal::None, internal::None,
01158     internal::None, internal::None, internal::None, internal::None,
01159     internal::None> {
01160     typedef internal::Types23<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01161         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23> type;
01162 };
01163 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01164     typename T6, typename T7, typename T8, typename T9, typename T10,
01165     typename T11, typename T12, typename T13, typename T14, typename T15,
01166     typename T16, typename T17, typename T18, typename T19, typename T20,
01167     typename T21, typename T22, typename T23, typename T24>
01168 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01169     T16, T17, T18, T19, T20, T21, T22, T23, T24, internal::None,
01170     internal::None, internal::None, internal::None, internal::None,
01171     internal::None, internal::None, internal::None, internal::None,
01172     internal::None, internal::None, internal::None, internal::None,
01173     internal::None, internal::None, internal::None, internal::None,
01174     internal::None, internal::None, internal::None, internal::None,
01175     internal::None, internal::None, internal::None, internal::None,
01176     internal::None> {
01177     typedef internal::Types24<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01178         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24> type;
01179 };
01180 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01181     typename T6, typename T7, typename T8, typename T9, typename T10,
01182     typename T11, typename T12, typename T13, typename T14, typename T15,
01183     typename T16, typename T17, typename T18, typename T19, typename T20,
01184     typename T21, typename T22, typename T23, typename T24, typename T25>
01185 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01186     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, internal::None,
01187     internal::None, internal::None, internal::None, internal::None,
01188     internal::None, internal::None, internal::None, internal::None,
01189     internal::None, internal::None, internal::None, internal::None,
01190     internal::None, internal::None, internal::None, internal::None,
01191     internal::None, internal::None, internal::None, internal::None,
01192     internal::None, internal::None, internal::None, internal::None> {
01193     typedef internal::Types25<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01194         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25> type;
01195 };
01196 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01197     typename T6, typename T7, typename T8, typename T9, typename T10,
01198     typename T11, typename T12, typename T13, typename T14, typename T15,
01199     typename T16, typename T17, typename T18, typename T19, typename T20,
01200     typename T21, typename T22, typename T23, typename T24, typename T25,
01201     typename T26>
01202 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01203     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, internal::None,
01204     internal::None, internal::None, internal::None, internal::None,
01205     internal::None, internal::None, internal::None, internal::None,
01206     internal::None, internal::None, internal::None, internal::None,
01207     internal::None, internal::None, internal::None, internal::None,
01208     internal::None, internal::None, internal::None, internal::None,
01209     internal::None, internal::None, internal::None> {
01210     typedef internal::Types26<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01211         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25,
01212         T26> type;
01213 };

```

```
01214 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01215     typename T6, typename T7, typename T8, typename T9, typename T10,
01216     typename T11, typename T12, typename T13, typename T14, typename T15,
01217     typename T16, typename T17, typename T18, typename T19, typename T20,
01218     typename T21, typename T22, typename T23, typename T24, typename T25,
01219     typename T26, typename T27>
01220 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01221     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, internal::None,
01222     internal::None, internal::None, internal::None, internal::None,
01223     internal::None, internal::None, internal::None, internal::None,
01224     internal::None, internal::None, internal::None, internal::None,
01225     internal::None, internal::None, internal::None, internal::None,
01226     internal::None, internal::None, internal::None, internal::None,
01227     internal::None, internal::None> {
01228     typedef internal::Types27<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01229         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01230         T27> type;
01231 };
01232 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01233     typename T6, typename T7, typename T8, typename T9, typename T10,
01234     typename T11, typename T12, typename T13, typename T14, typename T15,
01235     typename T16, typename T17, typename T18, typename T19, typename T20,
01236     typename T21, typename T22, typename T23, typename T24, typename T25,
01237     typename T26, typename T27, typename T28>
01238 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01239     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
01240     internal::None, internal::None, internal::None, internal::None,
01241     internal::None, internal::None, internal::None, internal::None,
01242     internal::None, internal::None, internal::None, internal::None,
01243     internal::None, internal::None, internal::None, internal::None,
01244     internal::None, internal::None, internal::None, internal::None,
01245     internal::None, internal::None> {
01246     typedef internal::Types28<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01247         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01248         T27, T28> type;
01249 };
01250 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01251     typename T6, typename T7, typename T8, typename T9, typename T10,
01252     typename T11, typename T12, typename T13, typename T14, typename T15,
01253     typename T16, typename T17, typename T18, typename T19, typename T20,
01254     typename T21, typename T22, typename T23, typename T24, typename T25,
01255     typename T26, typename T27, typename T28, typename T29>
01256 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01257     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
01258     internal::None, internal::None, internal::None, internal::None,
01259     internal::None, internal::None, internal::None, internal::None,
01260     internal::None, internal::None, internal::None, internal::None,
01261     internal::None, internal::None, internal::None, internal::None,
01262     internal::None, internal::None, internal::None, internal::None,
01263     internal::None> {
01264     typedef internal::Types29<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01265         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01266         T27, T28, T29> type;
01267 };
01268 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01269     typename T6, typename T7, typename T8, typename T9, typename T10,
01270     typename T11, typename T12, typename T13, typename T14, typename T15,
01271     typename T16, typename T17, typename T18, typename T19, typename T20,
01272     typename T21, typename T22, typename T23, typename T24, typename T25,
01273     typename T26, typename T27, typename T28, typename T29, typename T30>
01274 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01275     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01276     internal::None, internal::None, internal::None, internal::None,
01277     internal::None, internal::None, internal::None, internal::None,
01278     internal::None, internal::None, internal::None, internal::None,
01279     internal::None, internal::None, internal::None, internal::None,
01280     internal::None, internal::None, internal::None, internal::None> {
01281     typedef internal::Types30<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01282         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01283         T27, T28, T29, T30> type;
01284 };
01285 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01286     typename T6, typename T7, typename T8, typename T9, typename T10,
01287     typename T11, typename T12, typename T13, typename T14, typename T15,
01288     typename T16, typename T17, typename T18, typename T19, typename T20,
01289     typename T21, typename T22, typename T23, typename T24, typename T25,
01290     typename T26, typename T27, typename T28, typename T29, typename T30,
01291     typename T31>
01292 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01293     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01294     T31, internal::None, internal::None, internal::None, internal::None,
01295     internal::None, internal::None, internal::None, internal::None,
01296     internal::None, internal::None, internal::None, internal::None,
01297     internal::None, internal::None, internal::None, internal::None,
01298     internal::None, internal::None, internal::None> {
01299     typedef internal::Types31<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01300         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
```

```

01301     T27, T28, T29, T30, T31> type;
01302 };
01303 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01304     typename T6, typename T7, typename T8, typename T9, typename T10,
01305     typename T11, typename T12, typename T13, typename T14, typename T15,
01306     typename T16, typename T17, typename T18, typename T19, typename T20,
01307     typename T21, typename T22, typename T23, typename T24, typename T25,
01308     typename T26, typename T27, typename T28, typename T29, typename T30,
01309     typename T31, typename T32>
01310 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01311     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01312     T31, T32, internal::None, internal::None, internal::None, internal::None,
01313     internal::None, internal::None, internal::None, internal::None,
01314     internal::None, internal::None, internal::None, internal::None,
01315     internal::None, internal::None, internal::None, internal::None,
01316     internal::None, internal::None> {
01317     typedef internal::Types32<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01318         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01319         T27, T28, T29, T30, T31, T32> type;
01320 };
01321 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01322     typename T6, typename T7, typename T8, typename T9, typename T10,
01323     typename T11, typename T12, typename T13, typename T14, typename T15,
01324     typename T16, typename T17, typename T18, typename T19, typename T20,
01325     typename T21, typename T22, typename T23, typename T24, typename T25,
01326     typename T26, typename T27, typename T28, typename T29, typename T30,
01327     typename T31, typename T32, typename T33>
01328 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01329     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01330     T31, T32, T33, internal::None, internal::None, internal::None,
01331     internal::None, internal::None, internal::None, internal::None,
01332     internal::None, internal::None, internal::None, internal::None,
01333     internal::None, internal::None, internal::None, internal::None,
01334     internal::None, internal::None> {
01335     typedef internal::Types33<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01336         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01337         T27, T28, T29, T30, T31, T32, T33> type;
01338 };
01339 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01340     typename T6, typename T7, typename T8, typename T9, typename T10,
01341     typename T11, typename T12, typename T13, typename T14, typename T15,
01342     typename T16, typename T17, typename T18, typename T19, typename T20,
01343     typename T21, typename T22, typename T23, typename T24, typename T25,
01344     typename T26, typename T27, typename T28, typename T29, typename T30,
01345     typename T31, typename T32, typename T33, typename T34>
01346 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01347     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01348     T31, T32, T33, T34, internal::None, internal::None, internal::None,
01349     internal::None, internal::None, internal::None, internal::None,
01350     internal::None, internal::None, internal::None, internal::None,
01351     internal::None, internal::None, internal::None, internal::None,
01352     internal::None> {
01353     typedef internal::Types34<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01354         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01355         T27, T28, T29, T30, T31, T32, T33, T34> type;
01356 };
01357 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01358     typename T6, typename T7, typename T8, typename T9, typename T10,
01359     typename T11, typename T12, typename T13, typename T14, typename T15,
01360     typename T16, typename T17, typename T18, typename T19, typename T20,
01361     typename T21, typename T22, typename T23, typename T24, typename T25,
01362     typename T26, typename T27, typename T28, typename T29, typename T30,
01363     typename T31, typename T32, typename T33, typename T34, typename T35>
01364 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01365     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01366     T31, T32, T33, T34, T35, internal::None, internal::None, internal::None,
01367     internal::None, internal::None, internal::None, internal::None,
01368     internal::None, internal::None, internal::None, internal::None,
01369     internal::None, internal::None, internal::None, internal::None> {
01370     typedef internal::Types35<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01371         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01372         T27, T28, T29, T30, T31, T32, T33, T34, T35> type;
01373 };
01374 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01375     typename T6, typename T7, typename T8, typename T9, typename T10,
01376     typename T11, typename T12, typename T13, typename T14, typename T15,
01377     typename T16, typename T17, typename T18, typename T19, typename T20,
01378     typename T21, typename T22, typename T23, typename T24, typename T25,
01379     typename T26, typename T27, typename T28, typename T29, typename T30,
01380     typename T31, typename T32, typename T33, typename T34, typename T35,
01381     typename T36>
01382 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01383     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01384     T31, T32, T33, T34, T35, T36, internal::None, internal::None,
01385     internal::None, internal::None, internal::None, internal::None,
01386     internal::None, internal::None, internal::None, internal::None,
01387     internal::None, internal::None, internal::None> {

```

```
01388     typedef internal::Types36<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01389         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01390         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36> type;
01391 };
01392 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01393     typename T6, typename T7, typename T8, typename T9, typename T10,
01394     typename T11, typename T12, typename T13, typename T14, typename T15,
01395     typename T16, typename T17, typename T18, typename T19, typename T20,
01396     typename T21, typename T22, typename T23, typename T24, typename T25,
01397     typename T26, typename T27, typename T28, typename T29, typename T30,
01398     typename T31, typename T32, typename T33, typename T34, typename T35,
01399     typename T36, typename T37>
01400     struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01401         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01402         T31, T32, T33, T34, T35, T36, T37, internal::None, internal::None,
01403         internal::None, internal::None, internal::None, internal::None,
01404         internal::None, internal::None, internal::None, internal::None,
01405         internal::None, internal::None, internal::None> {
01406     typedef internal::Types37<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01407         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01408         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37> type;
01409 };
01410 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01411     typename T6, typename T7, typename T8, typename T9, typename T10,
01412     typename T11, typename T12, typename T13, typename T14, typename T15,
01413     typename T16, typename T17, typename T18, typename T19, typename T20,
01414     typename T21, typename T22, typename T23, typename T24, typename T25,
01415     typename T26, typename T27, typename T28, typename T29, typename T30,
01416     typename T31, typename T32, typename T33, typename T34, typename T35,
01417     typename T36, typename T37, typename T38>
01418     struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01419         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01420         T31, T32, T33, T34, T35, T36, T37, T38, internal::None, internal::None,
01421         internal::None, internal::None, internal::None, internal::None,
01422         internal::None, internal::None, internal::None, internal::None,
01423         internal::None, internal::None> {
01424     typedef internal::Types38<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01425         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01426         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38> type;
01427 };
01428 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01429     typename T6, typename T7, typename T8, typename T9, typename T10,
01430     typename T11, typename T12, typename T13, typename T14, typename T15,
01431     typename T16, typename T17, typename T18, typename T19, typename T20,
01432     typename T21, typename T22, typename T23, typename T24, typename T25,
01433     typename T26, typename T27, typename T28, typename T29, typename T30,
01434     typename T31, typename T32, typename T33, typename T34, typename T35,
01435     typename T36, typename T37, typename T38, typename T39>
01436     struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01437         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01438         T31, T32, T33, T34, T35, T36, T37, T38, T39, internal::None,
01439         internal::None, internal::None, internal::None, internal::None,
01440         internal::None, internal::None, internal::None, internal::None,
01441         internal::None, internal::None> {
01442     typedef internal::Types39<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01443         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01444         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38> type;
01445 };
01446 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01447     typename T6, typename T7, typename T8, typename T9, typename T10,
01448     typename T11, typename T12, typename T13, typename T14, typename T15,
01449     typename T16, typename T17, typename T18, typename T19, typename T20,
01450     typename T21, typename T22, typename T23, typename T24, typename T25,
01451     typename T26, typename T27, typename T28, typename T29, typename T30,
01452     typename T31, typename T32, typename T33, typename T34, typename T35,
01453     typename T36, typename T37, typename T38, typename T39, typename T40>
01454     struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01455         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01456         T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, internal::None,
01457         internal::None, internal::None, internal::None, internal::None,
01458         internal::None, internal::None, internal::None, internal::None,
01459         internal::None> {
01460     typedef internal::Types40<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01461         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01462         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39,
01463         T40> type;
01464 };
01465 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01466     typename T6, typename T7, typename T8, typename T9, typename T10,
01467     typename T11, typename T12, typename T13, typename T14, typename T15,
01468     typename T16, typename T17, typename T18, typename T19, typename T20,
01469     typename T21, typename T22, typename T23, typename T24, typename T25,
01470     typename T26, typename T27, typename T28, typename T29, typename T30,
01471     typename T31, typename T32, typename T33, typename T34, typename T35,
01472     typename T36, typename T37, typename T38, typename T39, typename T40,
01473     typename T41>
01474     struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
```

```

01475     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01476     T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, internal::None,
01477     internal::None, internal::None, internal::None, internal::None,
01478     internal::None, internal::None, internal::None, internal::None> {
01479     typedef internal::Types41<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01480         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01481         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40,
01482         T41> type;
01483 };
01484 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01485     typename T6, typename T7, typename T8, typename T9, typename T10,
01486     typename T11, typename T12, typename T13, typename T14, typename T15,
01487     typename T16, typename T17, typename T18, typename T19, typename T20,
01488     typename T21, typename T22, typename T23, typename T24, typename T25,
01489     typename T26, typename T27, typename T28, typename T29, typename T30,
01490     typename T31, typename T32, typename T33, typename T34, typename T35,
01491     typename T36, typename T37, typename T38, typename T39, typename T40,
01492     typename T41, typename T42>
01493     struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01494         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01495         T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, internal::None,
01496     internal::None, internal::None, internal::None, internal::None,
01497     internal::None, internal::None, internal::None> {
01498     typedef internal::Types42<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01499         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01500         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40,
01501         T41, T42> type;
01502 };
01503 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01504     typename T6, typename T7, typename T8, typename T9, typename T10,
01505     typename T11, typename T12, typename T13, typename T14, typename T15,
01506     typename T16, typename T17, typename T18, typename T19, typename T20,
01507     typename T21, typename T22, typename T23, typename T24, typename T25,
01508     typename T26, typename T27, typename T28, typename T29, typename T30,
01509     typename T31, typename T32, typename T33, typename T34, typename T35,
01510     typename T36, typename T37, typename T38, typename T39, typename T40,
01511     typename T41, typename T42, typename T43>
01512     struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01513         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01514         T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
01515     internal::None, internal::None, internal::None, internal::None,
01516     internal::None, internal::None, internal::None> {
01517     typedef internal::Types43<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01518         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01519         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40,
01520         T41, T42, T43> type;
01521 };
01522 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01523     typename T6, typename T7, typename T8, typename T9, typename T10,
01524     typename T11, typename T12, typename T13, typename T14, typename T15,
01525     typename T16, typename T17, typename T18, typename T19, typename T20,
01526     typename T21, typename T22, typename T23, typename T24, typename T25,
01527     typename T26, typename T27, typename T28, typename T29, typename T30,
01528     typename T31, typename T32, typename T33, typename T34, typename T35,
01529     typename T36, typename T37, typename T38, typename T39, typename T40,
01530     typename T41, typename T42, typename T43, typename T44>
01531     struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01532         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01533         T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44,
01534     internal::None, internal::None, internal::None, internal::None,
01535     internal::None, internal::None> {
01536     typedef internal::Types44<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01537         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01538         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40,
01539         T41, T42, T43, T44> type;
01540 };
01541 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01542     typename T6, typename T7, typename T8, typename T9, typename T10,
01543     typename T11, typename T12, typename T13, typename T14, typename T15,
01544     typename T16, typename T17, typename T18, typename T19, typename T20,
01545     typename T21, typename T22, typename T23, typename T24, typename T25,
01546     typename T26, typename T27, typename T28, typename T29, typename T30,
01547     typename T31, typename T32, typename T33, typename T34, typename T35,
01548     typename T36, typename T37, typename T38, typename T39, typename T40,
01549     typename T41, typename T42, typename T43, typename T44, typename T45>
01550     struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01551         T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01552         T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45,
01553     internal::None, internal::None, internal::None, internal::None,
01554     internal::None> {
01555     typedef internal::Types45<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01556         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01557         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40,
01558         T41, T42, T43, T44, T45> type;
01559 };
01560 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01561     typename T6, typename T7, typename T8, typename T9, typename T10,

```

```

01562     typename T11, typename T12, typename T13, typename T14, typename T15,
01563     typename T16, typename T17, typename T18, typename T19, typename T20,
01564     typename T21, typename T22, typename T23, typename T24, typename T25,
01565     typename T26, typename T27, typename T28, typename T29, typename T30,
01566     typename T31, typename T32, typename T33, typename T34, typename T35,
01567     typename T36, typename T37, typename T38, typename T39, typename T40,
01568     typename T41, typename T42, typename T43, typename T44, typename T45,
01569     typename T46>
01570 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01571     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01572     T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45,
01573     T46, internal::None, internal::None, internal::None, internal::None> {
01574     typedef internal::Types46<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01575         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01576         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40,
01577         T41, T42, T43, T44, T45, T46> type;
01578 };
01579 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01580     typename T6, typename T7, typename T8, typename T9, typename T10,
01581     typename T11, typename T12, typename T13, typename T14, typename T15,
01582     typename T16, typename T17, typename T18, typename T19, typename T20,
01583     typename T21, typename T22, typename T23, typename T24, typename T25,
01584     typename T26, typename T27, typename T28, typename T29, typename T30,
01585     typename T31, typename T32, typename T33, typename T34, typename T35,
01586     typename T36, typename T37, typename T38, typename T39, typename T40,
01587     typename T41, typename T42, typename T43, typename T44, typename T45,
01588     typename T46, typename T47>
01589 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01590     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01591     T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45,
01592     T46, T47, internal::None, internal::None, internal::None> {
01593     typedef internal::Types47<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01594         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01595         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40,
01596         T41, T42, T43, T44, T45, T46, T47> type;
01597 };
01598 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01599     typename T6, typename T7, typename T8, typename T9, typename T10,
01600     typename T11, typename T12, typename T13, typename T14, typename T15,
01601     typename T16, typename T17, typename T18, typename T19, typename T20,
01602     typename T21, typename T22, typename T23, typename T24, typename T25,
01603     typename T26, typename T27, typename T28, typename T29, typename T30,
01604     typename T31, typename T32, typename T33, typename T34, typename T35,
01605     typename T36, typename T37, typename T38, typename T39, typename T40,
01606     typename T41, typename T42, typename T43, typename T44, typename T45,
01607     typename T46, typename T47, typename T48>
01608 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01609     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01610     T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45,
01611     T46, T47, T48, internal::None, internal::None> {
01612     typedef internal::Types48<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01613         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01614         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40,
01615         T41, T42, T43, T44, T45, T46, T47, T48> type;
01616 };
01617 template <typename T1, typename T2, typename T3, typename T4, typename T5,
01618     typename T6, typename T7, typename T8, typename T9, typename T10,
01619     typename T11, typename T12, typename T13, typename T14, typename T15,
01620     typename T16, typename T17, typename T18, typename T19, typename T20,
01621     typename T21, typename T22, typename T23, typename T24, typename T25,
01622     typename T26, typename T27, typename T28, typename T29, typename T30,
01623     typename T31, typename T32, typename T33, typename T34, typename T35,
01624     typename T36, typename T37, typename T38, typename T39, typename T40,
01625     typename T41, typename T42, typename T43, typename T44, typename T45,
01626     typename T46, typename T47, typename T48, typename T49>
01627 struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
01628     T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30,
01629     T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44, T45,
01630     T46, T47, T48, T49, internal::None> {
01631     typedef internal::Types49<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
01632         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
01633         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40,
01634         T41, T42, T43, T44, T45, T46, T47, T48, T49> type;
01635 };
01636
01637 namespace internal {
01638
01639 # define GTEST_TEMPLATE_ template <typename T> class
01640
01641 // The template "selector" struct TemplateSel<Tmpl> is used to
01642 // represent Tmpl, which must be a class template with one type
01643 // parameter, as a type. TemplateSel<Tmpl>::Bind<T>::type is defined
01644 // as the type Tmpl<T>. This allows us to actually instantiate the
01645 // template "selected" by TemplateSel<Tmpl>.
01646 //
01647 // This trick is necessary for simulating typeid for class templates,
01648 // which C++ doesn't support directly.

```

```
01649 template <GTEST_TEMPLATE_ Tmpl>
01650 struct TemplateSel {
01651     template <typename T>
01652     struct Bind {
01653         typedef Tmpl<T> type;
01654     };
01655 };
01656
01657 # define GTEST_BIND_(TmplSel, T) \
01658     TmplSel::template Bind<T>::type
01659
01660 // A unique struct template used as the default value for the
01661 // arguments of class template Templates. This allows us to simulate
01662 // variadic templates (e.g. Templates<int>, Templates<int, double>,
01663 // and etc), which C++ doesn't support directly.
01664 template <typename T>
01665 struct NoneT {};
01666
01667 // The following family of struct and struct templates are used to
01668 // represent template lists. In particular, TemplatesN<T1, T2, ...,
01669 // TN> represents a list of N templates (T1, T2, ..., and TN). Except
01670 // for Templates0, every struct in the family has two member types:
01671 // Head for the selector of the first template in the list, and Tail
01672 // for the rest of the list.
01673
01674 // The empty template list.
01675 struct Templates0 {};
01676
01677 // Template lists of length 1, 2, 3, and so on.
01678
01679 template <GTEST_TEMPLATE_ T1>
01680 struct Templates1 {
01681     typedef TemplateSel<T1> Head;
01682     typedef Templates0 Tail;
01683 };
01684 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2>
01685 struct Templates2 {
01686     typedef TemplateSel<T1> Head;
01687     typedef Templates1<T2> Tail;
01688 };
01689
01690 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3>
01691 struct Templates3 {
01692     typedef TemplateSel<T1> Head;
01693     typedef Templates2<T2, T3> Tail;
01694 };
01695
01696 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01697           GTEST_TEMPLATE_ T4>
01698 struct Templates4 {
01699     typedef TemplateSel<T1> Head;
01700     typedef Templates3<T2, T3, T4> Tail;
01701 };
01702
01703 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01704           GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5>
01705 struct Templates5 {
01706     typedef TemplateSel<T1> Head;
01707     typedef Templates4<T2, T3, T4, T5> Tail;
01708 };
01709
01710 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01711           GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6>
01712 struct Templates6 {
01713     typedef TemplateSel<T1> Head;
01714     typedef Templates5<T2, T3, T4, T5, T6> Tail;
01715 };
01716
01717 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01718           GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01719           GTEST_TEMPLATE_ T7>
01720 struct Templates7 {
01721     typedef TemplateSel<T1> Head;
01722     typedef Templates6<T2, T3, T4, T5, T6, T7> Tail;
01723 };
01724
01725 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01726           GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01727           GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8>
01728 struct Templates8 {
01729     typedef TemplateSel<T1> Head;
01730     typedef Templates7<T2, T3, T4, T5, T6, T7, T8> Tail;
01731 };
01732
01733 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01734           GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01735           GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9>
```

```
01736 struct Templates9 {
01737     typedef TemplateSel<T1> Head;
01738     typedef Templates8<T2, T3, T4, T5, T6, T7, T8, T9> Tail;
01739 };
01740
01741 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01742             GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01743             GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01744             GTEST_TEMPLATE_ T10>
01745 struct Templates10 {
01746     typedef TemplateSel<T1> Head;
01747     typedef Templates9<T2, T3, T4, T5, T6, T7, T8, T9, T10> Tail;
01748 };
01749
01750 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01751             GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01752             GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01753             GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11>
01754 struct Templates11 {
01755     typedef TemplateSel<T1> Head;
01756     typedef Templates10<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11> Tail;
01757 };
01758
01759 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01760             GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01761             GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01762             GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12>
01763 struct Templates12 {
01764     typedef TemplateSel<T1> Head;
01765     typedef Templates11<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12> Tail;
01766 };
01767
01768 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01769             GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01770             GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01771             GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01772             GTEST_TEMPLATE_ T13>
01773 struct Templates13 {
01774     typedef TemplateSel<T1> Head;
01775     typedef Templates12<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13> Tail;
01776 };
01777
01778 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01779             GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01780             GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01781             GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01782             GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14>
01783 struct Templates14 {
01784     typedef TemplateSel<T1> Head;
01785     typedef Templates13<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
01786                         T14> Tail;
01787 };
01788
01789 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01790             GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01791             GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01792             GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01793             GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15>
01794 struct Templates15 {
01795     typedef TemplateSel<T1> Head;
01796     typedef Templates14<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01797                         T15> Tail;
01798 };
01799
01800 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01801             GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01802             GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01803             GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01804             GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
01805             GTEST_TEMPLATE_ T16>
01806 struct Templates16 {
01807     typedef TemplateSel<T1> Head;
01808     typedef Templates15<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01809                         T15, T16> Tail;
01810 };
01811
01812 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01813             GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01814             GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01815             GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01816             GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
01817             GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17>
01818 struct Templates17 {
01819     typedef TemplateSel<T1> Head;
01820     typedef Templates16<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01821                         T15, T16, T17> Tail;
01822 };
```

```

01823 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01824     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01825     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01826     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01827     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
01828     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18>
01829 struct Templates18 {
01830     typedef TemplateSel<T1> Head;
01831     typedef Templates17<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01832             T15, T16, T17, T18> Tail;
01833 };
01834 };
01835
01836 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01837     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01838     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01839     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01840     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
01841     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
01842     GTEST_TEMPLATE_ T19>
01843 struct Templates19 {
01844     typedef TemplateSel<T1> Head;
01845     typedef Templates18<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01846             T15, T16, T17, T18, T19> Tail;
01847 };
01848
01849 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01850     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01851     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01852     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01853     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
01854     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
01855     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20>
01856 struct Templates20 {
01857     typedef TemplateSel<T1> Head;
01858     typedef Templates19<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01859             T15, T16, T17, T18, T19, T20> Tail;
01860 };
01861
01862 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01863     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01864     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01865     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01866     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
01867     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
01868     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21>
01869 struct Templates21 {
01870     typedef TemplateSel<T1> Head;
01871     typedef Templates20<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01872             T15, T16, T17, T18, T19, T20, T21> Tail;
01873 };
01874
01875 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01876     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01877     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01878     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01879     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
01880     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
01881     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
01882     GTEST_TEMPLATE_ T22>
01883 struct Templates22 {
01884     typedef TemplateSel<T1> Head;
01885     typedef Templates21<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01886             T15, T16, T17, T18, T19, T20, T21, T22> Tail;
01887 };
01888
01889 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01890     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01891     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01892     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01893     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
01894     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
01895     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
01896     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23>
01897 struct Templates23 {
01898     typedef TemplateSel<T1> Head;
01899     typedef Templates22<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01900             T15, T16, T17, T18, T19, T20, T21, T22, T23> Tail;
01901 };
01902
01903 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01904     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01905     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01906     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01907     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
01908     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
01909     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,

```

```
01910     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24>
01911 struct Templates24 {
01912     typedef TemplateSel<T1> Head;
01913     typedef Templates23<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01914         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24> Tail;
01915 };
01916
01917 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01918     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01919     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01920     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01921     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
01922     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
01923     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
01924     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
01925     GTEST_TEMPLATE_ T25>
01926 struct Templates25 {
01927     typedef TemplateSel<T1> Head;
01928     typedef Templates24<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01929         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25> Tail;
01930 };
01931
01932 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01933     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01934     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01935     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01936     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
01937     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
01938     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
01939     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
01940     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26>
01941 struct Templates26 {
01942     typedef TemplateSel<T1> Head;
01943     typedef Templates25<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01944         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26> Tail;
01945 };
01946
01947 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01948     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01949     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01950     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01951     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
01952     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
01953     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
01954     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
01955     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27>
01956 struct Templates27 {
01957     typedef TemplateSel<T1> Head;
01958     typedef Templates26<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01959         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27> Tail;
01960 };
01961
01962 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01963     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01964     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01965     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01966     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
01967     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
01968     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
01969     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
01970     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
01971     GTEST_TEMPLATE_ T28>
01972 struct Templates28 {
01973     typedef TemplateSel<T1> Head;
01974     typedef Templates27<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01975         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
01976         T28> Tail;
01977 };
01978
01979 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
01980     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01981     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01982     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
01983     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
01984     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
01985     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
01986     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
01987     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
01988     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29>
01989 struct Templates29 {
01990     typedef TemplateSel<T1> Head;
01991     typedef Templates28<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
01992         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
01993         T29> Tail;
01994 };
01995
01996 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
```

```

01997     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
01998     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
01999     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02000     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02001     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02002     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02003     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02004     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02005     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30>
02006 struct Templates30 {
02007     typedef TemplateSel<T1> Head;
02008     typedef Templates29<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02009         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02010         T29, T30> Tail;
02011 };
02012
02013 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02014     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02015     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02016     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02017     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02018     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02019     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02020     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02021     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02022     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02023     GTEST_TEMPLATE_ T31>
02024 struct Templates31 {
02025     typedef TemplateSel<T1> Head;
02026     typedef Templates30<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02027         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02028         T29, T30, T31> Tail;
02029 };
02030
02031 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02032     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02033     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02034     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02035     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02036     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02037     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02038     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02039     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02040     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02041     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32>
02042 struct Templates32 {
02043     typedef TemplateSel<T1> Head;
02044     typedef Templates31<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02045         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02046         T29, T30, T31, T32> Tail;
02047 };
02048
02049 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02050     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02051     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02052     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02053     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02054     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02055     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02056     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02057     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02058     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02059     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33>
02060 struct Templates33 {
02061     typedef TemplateSel<T1> Head;
02062     typedef Templates32<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02063         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02064         T29, T30, T31, T32, T33> Tail;
02065 };
02066
02067 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02068     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02069     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02070     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02071     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02072     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02073     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02074     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02075     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02076     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02077     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02078     GTEST_TEMPLATE_ T34>
02079 struct Templates34 {
02080     typedef TemplateSel<T1> Head;
02081     typedef Templates33<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02082         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02083         T29, T30, T31, T32, T33, T34> Tail;

```

```
02084 };
02085
02086 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02087     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02088     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02089     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02090     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02091     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02092     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02093     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02094     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02095     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02096     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02097     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35>
02098 struct Templates35 {
02099     typedef TemplateSel<T1> Head;
02100     typedef Templates34<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02101         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02102         T29, T30, T31, T32, T33, T34, T35> Tail;
02103 };
02104
02105 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02106     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02107     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02108     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02109     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02110     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02111     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02112     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02113     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02114     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02115     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02116     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36>
02117 struct Templates36 {
02118     typedef TemplateSel<T1> Head;
02119     typedef Templates35<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02120         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02121         T29, T30, T31, T32, T33, T34, T35, T36> Tail;
02122 };
02123
02124 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02125     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02126     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02127     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02128     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02129     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02130     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02131     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02132     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02133     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02134     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02135     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
02136     GTEST_TEMPLATE_ T37>
02137 struct Templates37 {
02138     typedef TemplateSel<T1> Head;
02139     typedef Templates36<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02140         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02141         T29, T30, T31, T32, T33, T34, T35, T36, T37> Tail;
02142 };
02143
02144 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02145     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02146     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02147     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02148     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02149     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02150     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02151     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02152     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02153     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02154     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02155     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
02156     GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38>
02157 struct Templates38 {
02158     typedef TemplateSel<T1> Head;
02159     typedef Templates37<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02160         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02161         T29, T30, T31, T32, T33, T34, T35, T36, T37, T38> Tail;
02162 };
02163
02164 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02165     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02166     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02167     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02168     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02169     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02170     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
```

```

02171     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02172     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02173     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02174     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02175     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
02176     GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39>
02177 struct Templates39 {
02178     typedef TemplateSel<T1> Head;
02179     typedef Templates38<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02180             T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02181             T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39> Tail;
02182 };
02183
02184 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02185         GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02186         GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02187         GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02188         GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02189         GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02190         GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02191         GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02192         GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02193         GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02194         GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02195         GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
02196         GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
02197         GTEST_TEMPLATE_ T40>
02198 struct Templates40 {
02199     typedef TemplateSel<T1> Head;
02200     typedef Templates39<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02201             T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02202             T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40> Tail;
02203 };
02204
02205 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02206         GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02207         GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02208         GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02209         GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02210         GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02211         GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02212         GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02213         GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02214         GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02215         GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02216         GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
02217         GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
02218         GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41>
02219 struct Templates41 {
02220     typedef TemplateSel<T1> Head;
02221     typedef Templates40<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02222             T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02223             T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41> Tail;
02224 };
02225
02226 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02227         GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02228         GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02229         GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02230         GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02231         GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02232         GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02233         GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02234         GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02235         GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02236         GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02237         GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
02238         GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
02239         GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42>
02240 struct Templates42 {
02241     typedef TemplateSel<T1> Head;
02242     typedef Templates41<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02243             T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02244             T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41,
02245             T42> Tail;
02246 };
02247
02248 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02249         GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02250         GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02251         GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02252         GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02253         GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02254         GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02255         GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02256         GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02257         GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,

```

```
02258     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02259     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
02260     GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
02261     GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
02262     GTEST_TEMPLATE_ T43>
02263 struct Templates43 {
02264     typedef TemplateSel<T1> Head;
02265     typedef Templates42<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02266         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02267         T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42,
02268         T43> Tail;
02269 };
02270
02271 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02272     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02273     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02274     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02275     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02276     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02277     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02278     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02279     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02280     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02281     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02282     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
02283     GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
02284     GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
02285     GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44>
02286 struct Templates44 {
02287     typedef TemplateSel<T1> Head;
02288     typedef Templates43<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02289         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02290         T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42,
02291         T43, T44> Tail;
02292 };
02293
02294 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02295     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02296     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02297     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02298     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02299     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02300     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02301     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02302     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02303     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02304     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02305     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
02306     GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
02307     GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
02308     GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45>
02309 struct Templates45 {
02310     typedef TemplateSel<T1> Head;
02311     typedef Templates44<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02312         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02313         T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42,
02314         T43, T44, T45> Tail;
02315 };
02316
02317 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02318     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02319     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02320     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02321     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02322     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02323     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02324     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02325     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02326     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02327     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02328     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
02329     GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
02330     GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
02331     GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
02332     GTEST_TEMPLATE_ T46>
02333 struct Templates46 {
02334     typedef TemplateSel<T1> Head;
02335     typedef Templates45<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02336         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02337         T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42,
02338         T43, T44, T45, T46> Tail;
02339 };
02340
02341 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02342     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02343     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02344     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
```

```

02345     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02346     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02347     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02348     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02349     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02350     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02351     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02352     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
02353     GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
02354     GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
02355     GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
02356     GTEST_TEMPLATE_ T46, GTEST_TEMPLATE_ T47>
02357 struct Templates47 {
02358     typedef TemplateSel<T1> Head;
02359     typedef Templates46<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02360         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02361         T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42,
02362         T43, T44, T45, T46, T47> Tail;
02363 };
02364
02365 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02366     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02367     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02368     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02369     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02370     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02371     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02372     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02373     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02374     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02375     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02376     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
02377     GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
02378     GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
02379     GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
02380     GTEST_TEMPLATE_ T46, GTEST_TEMPLATE_ T47, GTEST_TEMPLATE_ T48>
02381 struct Templates48 {
02382     typedef TemplateSel<T1> Head;
02383     typedef Templates47<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02384         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02385         T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42,
02386         T43, T44, T45, T46, T47, T48> Tail;
02387 };
02388
02389 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02390     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02391     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02392     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02393     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02394     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02395     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02396     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02397     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02398     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02399     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02400     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
02401     GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
02402     GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
02403     GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
02404     GTEST_TEMPLATE_ T46, GTEST_TEMPLATE_ T47, GTEST_TEMPLATE_ T48,
02405     GTEST_TEMPLATE_ T49>
02406 struct Templates49 {
02407     typedef TemplateSel<T1> Head;
02408     typedef Templates48<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02409         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02410         T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42,
02411         T43, T44, T45, T46, T47, T48> Tail;
02412 };
02413
02414 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02415     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02416     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02417     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02418     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02419     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02420     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02421     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02422     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02423     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02424     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02425     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
02426     GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
02427     GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
02428     GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
02429     GTEST_TEMPLATE_ T46, GTEST_TEMPLATE_ T47, GTEST_TEMPLATE_ T48,
02430     GTEST_TEMPLATE_ T49, GTEST_TEMPLATE_ T50>
02431 struct Templates50 {

```

```

02432     typedef TemplateSel<T1> Head;
02433     typedef Templates49<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02434         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02435         T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42,
02436         T43, T44, T45, T46, T47, T48, T49, T50> Tail;
02437 };
02438
02439
02440 // We don't want to require the users to write TemplatesN<...> directly,
02441 // as that would require them to count the length. Templates<...> is much
02442 // easier to write, but generates horrible messages when there is a
02443 // compiler error, as gcc insists on printing out each template
02444 // argument, even if it has the default value (this means Templates<list>
02445 // will appear as Templates<list, NoneT, NoneT, ..., NoneT> in the compiler
02446 // errors).
02447 //
02448 // Our solution is to combine the best part of the two approaches: a
02449 // user would write Templates<T1, ..., TN>, and Google Test will translate
02450 // that to TemplatesN<T1, ..., TN> internally to make error messages
02451 // readable. The translation is done by the 'type' member of the
02452 // Templates template.
02453 template <GTEST_TEMPLATE_ T1 = NoneT, GTEST_TEMPLATE_ T2 = NoneT,
02454     GTEST_TEMPLATE_ T3 = NoneT, GTEST_TEMPLATE_ T4 = NoneT,
02455     GTEST_TEMPLATE_ T5 = NoneT, GTEST_TEMPLATE_ T6 = NoneT,
02456     GTEST_TEMPLATE_ T7 = NoneT, GTEST_TEMPLATE_ T8 = NoneT,
02457     GTEST_TEMPLATE_ T9 = NoneT, GTEST_TEMPLATE_ T10 = NoneT,
02458     GTEST_TEMPLATE_ T11 = NoneT, GTEST_TEMPLATE_ T12 = NoneT,
02459     GTEST_TEMPLATE_ T13 = NoneT, GTEST_TEMPLATE_ T14 = NoneT,
02460     GTEST_TEMPLATE_ T15 = NoneT, GTEST_TEMPLATE_ T16 = NoneT,
02461     GTEST_TEMPLATE_ T17 = NoneT, GTEST_TEMPLATE_ T18 = NoneT,
02462     GTEST_TEMPLATE_ T19 = NoneT, GTEST_TEMPLATE_ T20 = NoneT,
02463     GTEST_TEMPLATE_ T21 = NoneT, GTEST_TEMPLATE_ T22 = NoneT,
02464     GTEST_TEMPLATE_ T23 = NoneT, GTEST_TEMPLATE_ T24 = NoneT,
02465     GTEST_TEMPLATE_ T25 = NoneT, GTEST_TEMPLATE_ T26 = NoneT,
02466     GTEST_TEMPLATE_ T27 = NoneT, GTEST_TEMPLATE_ T28 = NoneT,
02467     GTEST_TEMPLATE_ T29 = NoneT, GTEST_TEMPLATE_ T30 = NoneT,
02468     GTEST_TEMPLATE_ T31 = NoneT, GTEST_TEMPLATE_ T32 = NoneT,
02469     GTEST_TEMPLATE_ T33 = NoneT, GTEST_TEMPLATE_ T34 = NoneT,
02470     GTEST_TEMPLATE_ T35 = NoneT, GTEST_TEMPLATE_ T36 = NoneT,
02471     GTEST_TEMPLATE_ T37 = NoneT, GTEST_TEMPLATE_ T38 = NoneT,
02472     GTEST_TEMPLATE_ T39 = NoneT, GTEST_TEMPLATE_ T40 = NoneT,
02473     GTEST_TEMPLATE_ T41 = NoneT, GTEST_TEMPLATE_ T42 = NoneT,
02474     GTEST_TEMPLATE_ T43 = NoneT, GTEST_TEMPLATE_ T44 = NoneT,
02475     GTEST_TEMPLATE_ T45 = NoneT, GTEST_TEMPLATE_ T46 = NoneT,
02476     GTEST_TEMPLATE_ T47 = NoneT, GTEST_TEMPLATE_ T48 = NoneT,
02477     GTEST_TEMPLATE_ T49 = NoneT, GTEST_TEMPLATE_ T50 = NoneT>
02478     struct Templates {
02479     typedef Templates50<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02480         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
02481         T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41,
02482         T42, T43, T44, T45, T46, T47, T48, T49, T50> type;
02483 };
02484
02485 template <>
02486     struct Templates<NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02487         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02488         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02489         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02490         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02491         NoneT> {
02492     typedef Templates0 type;
02493 };
02494 template <GTEST_TEMPLATE_ T1>
02495     struct Templates<T1, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02496         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02497         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02498         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02499         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02500         NoneT> {
02501     typedef Templates1<T1> type;
02502 };
02503 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2>
02504     struct Templates<T1, T2, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02505         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02506         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02507         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02508         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02509         NoneT> {
02510     typedef Templates2<T1, T2> type;
02511 };
02512 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3>
02513     struct Templates<T1, T2, T3, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02514         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02515         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02516         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02517         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02518     typedef Templates3<T1, T2, T3> type;

```

```

02519 };
02520 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02521     GTEST_TEMPLATE_ T4>
02522 struct Templates<T1, T2, T3, T4, NoneT, NoneT, NoneT, NoneT, NoneT,
02523     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02524     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02525     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02526     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02527     typedef Templates4<T1, T2, T3, T4> type;
02528 };
02529 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02530     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5>
02531 struct Templates<T1, T2, T3, T4, T5, NoneT, NoneT, NoneT, NoneT, NoneT,
02532     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02533     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02534     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02535     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02536     typedef Templates5<T1, T2, T3, T4, T5> type;
02537 };
02538 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02539     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6>
02540 struct Templates<T1, T2, T3, T4, T5, T6, NoneT, NoneT, NoneT, NoneT,
02541     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02542     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02543     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02544     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02545     typedef Templates6<T1, T2, T3, T4, T5, T6> type;
02546 };
02547 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02548     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02549     GTEST_TEMPLATE_ T7>
02550 struct Templates<T1, T2, T3, T4, T5, T6, T7, NoneT, NoneT, NoneT, NoneT,
02551     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02552     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02553     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02554     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02555     typedef Templates7<T1, T2, T3, T4, T5, T6, T7> type;
02556 };
02557 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02558     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02559     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8>
02560 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, NoneT, NoneT, NoneT, NoneT,
02561     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02562     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02563     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02564     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02565     typedef Templates8<T1, T2, T3, T4, T5, T6, T7, T8> type;
02566 };
02567 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02568     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02569     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9>
02570 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, NoneT, NoneT, NoneT,
02571     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02572     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02573     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02574     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02575     typedef Templates9<T1, T2, T3, T4, T5, T6, T7, T8, T9> type;
02576 };
02577 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02578     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02579     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02580     GTEST_TEMPLATE_ T10>
02581 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, NoneT, NoneT, NoneT,
02582     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02583     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02584     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02585     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02586     typedef Templates10<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10> type;
02587 };
02588 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02589     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02590     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02591     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11>
02592 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, NoneT, NoneT,
02593     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02594     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02595     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02596     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02597     typedef Templates11<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11> type;
02598 };
02599 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02600     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02601     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02602     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12>
02603 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, NoneT,
02604     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02605     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT>

```

```
02606     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02607     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02608     typedef Templates12<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12> type;
02609 };
02610 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02611     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02612     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02613     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02614     GTEST_TEMPLATE_ T13>
02615     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, NoneT,
02616     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02617     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02618     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02619     NoneT, NoneT, NoneT, NoneT, NoneT> {
02620     typedef Templates13<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
02621     T13> type;
02622 };
02623 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02624     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02625     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02626     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02627     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14>
02628     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02629     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02630     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02631     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02632     NoneT, NoneT, NoneT, NoneT, NoneT> {
02633     typedef Templates14<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02634     T14> type;
02635 };
02636 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02637     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02638     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02639     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02640     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15>
02641     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02642     T15, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02643     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02644     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02645     NoneT, NoneT, NoneT, NoneT> {
02646     typedef Templates15<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02647     T14, T15> type;
02648 };
02649 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02650     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02651     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02652     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02653     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02654     GTEST_TEMPLATE_ T16>
02655     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02656     T15, T16, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02657     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02658     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02659     NoneT, NoneT, NoneT, NoneT> {
02660     typedef Templates16<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02661     T14, T15, T16> type;
02662 };
02663 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02664     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02665     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02666     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02667     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02668     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17>
02669     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
0270     T15, T16, T17, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
0271     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
0272     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
0273     NoneT, NoneT, NoneT, NoneT> {
0274     typedef Templates17<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
0275     T14, T15, T16, T17> type;
0276 };
0277 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
0278     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
0279     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
0280     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
0281     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
0282     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18>
0283     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
0284     T15, T16, T17, T18, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
0285     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
0286     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
0287     NoneT, NoneT, NoneT, NoneT> {
0288     typedef Templates18<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
0289     T14, T15, T16, T17, T18> type;
0290 };
0291 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
0292     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
```

```

02693     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02694     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02695     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02696     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02697     GTEST_TEMPLATE_ T19>
02698 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02699     T15, T16, T17, T18, T19, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02700     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02701     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02702     NoneT, NoneT, NoneT, NoneT> {
02703     typedef Templates19<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02704     T14, T15, T16, T17, T18, T19> type;
02705 };
02706 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02707     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02708     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02709     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02710     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02711     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02712     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20>
02713 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02714     T15, T16, T17, T18, T19, T20, NoneT, NoneT, NoneT, NoneT, NoneT,
02715     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02716     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02717     NoneT, NoneT, NoneT> {
02718     typedef Templates20<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02719     T14, T15, T16, T17, T18, T19, T20> type;
02720 };
02721 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02722     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02723     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02724     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02725     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02726     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02727     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21>
02728 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02729     T15, T16, T17, T18, T19, T20, T21, NoneT, NoneT, NoneT, NoneT, NoneT,
02730     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02731     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02732     NoneT, NoneT, NoneT> {
02733     typedef Templates21<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02734     T14, T15, T16, T17, T18, T19, T20, T21> type;
02735 };
02736 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02737     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02738     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02739     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02740     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02741     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02742     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02743     GTEST_TEMPLATE_ T22>
02744 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02745     T15, T16, T17, T18, T19, T20, T21, T22, NoneT, NoneT, NoneT, NoneT, NoneT,
02746     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02747     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02748     NoneT, NoneT, NoneT> {
02749     typedef Templates22<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02750     T14, T15, T16, T17, T18, T19, T20, T21, T22> type;
02751 };
02752 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02753     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02754     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02755     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02756     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02757     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02758     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02759     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23>
02760 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02761     T15, T16, T17, T18, T19, T20, T21, T22, T23, NoneT, NoneT, NoneT, NoneT,
02762     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02763     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02764     NoneT, NoneT, NoneT> {
02765     typedef Templates23<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02766     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23> type;
02767 };
02768 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02769     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02770     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02771     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02772     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02773     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02774     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02775     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24>
02776 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02777     T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, NoneT, NoneT, NoneT, NoneT,
02778     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02779     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT>
```

```
02780     NoneT, NoneT, NoneT> {
02781     typedef Templates24<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02782         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24> type;
02783 };
02784 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02785     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02786     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02787     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02788     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02789     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02790     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02791     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02792     GTEST_TEMPLATE_ T25>
02793     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02794         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25> type;
02795     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02796     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02797     NoneT, NoneT> {
02798     typedef Templates25<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02799         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25> type;
02800 };
02801 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02802     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02803     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02804     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02805     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02806     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02807     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02808     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02809     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26>
02810     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02811         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26> type;
02812     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02813     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02814     NoneT, NoneN> {
02815     typedef Templates26<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02816         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26> type;
02817 };
02818 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02819     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02820     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02821     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02822     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02823     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02824     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02825     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02826     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27>
02827     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02828         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, NoneT,
02829         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02830         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02831         NoneT, NoneN> {
02832     typedef Templates27<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02833         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
02834         T27> type;
02835 };
02836 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02837     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02838     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02839     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02840     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02841     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02842     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02843     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02844     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27>
02845     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02846         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
02847         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02848         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02849         NoneT, NoneN> {
02850     typedef Templates28<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02851         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
02852         T28> type;
02853 };
02854 };
02855 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02856     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02857     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02858     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02859     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02860     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02861     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02862     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02863     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02864     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29>
02865     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02866         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
```

```

02867     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02868     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02869     NoneT> {
02870     typedef Templates29<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02871         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
02872         T28, T29> type;
02873 };
02874 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02875     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02876     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02877     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02878     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02879     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02880     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02881     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02882     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02883     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30>
02884 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02885     T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
02886     T30, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02887     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02888     typedef Templates30<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02889         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
02890         T28, T29, T30> type;
02891 };
02892 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02893     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02894     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02895     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02896     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02897     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02898     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02899     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02900     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02901     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02902     GTEST_TEMPLATE_ T31>
02903 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02904     T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
02905     T30, T31, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02906     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02907     typedef Templates31<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02908         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
02909         T28, T29, T30, T31> type;
02910 };
02911 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02912     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02913     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02914     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02915     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02916     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02917     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02918     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02919     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02920     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02921     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32>
02922 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02923     T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
02924     T30, T31, T32, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02925     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02926     typedef Templates32<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02927         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
02928         T28, T29, T30, T31, T32> type;
02929 };
02930 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02931     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02932     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02933     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02934     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02935     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02936     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02937     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02938     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02939     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02940     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33>
02941 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02942     T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
02943     T30, T31, T32, T33, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02944     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02945     typedef Templates33<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02946         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
02947         T28, T29, T30, T31, T32, T33> type;
02948 };
02949 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02950     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02951     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02952     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02953     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,

```

```
02954     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02955     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02956     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02957     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02958     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02959     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02960     GTEST_TEMPLATE_ T34>
02961     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02962         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
02963         T30, T31, T32, T33, T34, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02964         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02965     typedef Templates34<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02966         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
02967         T28, T29, T30, T31, T32, T33, T34> type;
02968 };
02969 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02970     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02971     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02972     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02973     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02974     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02975     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02976     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02977     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02978     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02979     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
02980     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35>
02981     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
02982         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
02983         T30, T31, T32, T33, T34, T35, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
02984         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
02985     typedef Templates35<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
02986         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
02987         T28, T29, T30, T31, T32, T33, T34, T35> type;
02988 };
02989 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
02990     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
02991     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
02992     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
02993     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
02994     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
02995     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
02996     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
02997     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
02998     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
02999     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
03000     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36>
03001     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
03002         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
03003         T30, T31, T32, T33, T34, T35, T36, NoneT, NoneT, NoneT, NoneT, NoneT,
03004         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
03005     typedef Templates36<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
03006         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
03007         T28, T29, T30, T31, T32, T33, T34, T35, T36> type;
03008 };
03009 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
03010     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
03011     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
03012     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
03013     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
03014     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
03015     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
03016     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
03017     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
03018     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
03019     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
03020     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
03021     GTEST_TEMPLATE_ T37>
03022     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
03023         T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
03024         T30, T31, T32, T33, T34, T35, T36, T37, NoneT, NoneT, NoneT, NoneT, NoneT,
03025         NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
03026     typedef Templates37<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
03027         T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
03028         T28, T29, T30, T31, T32, T33, T34, T35, T36, T37> type;
03029 };
03030 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
03031     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
03032     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
03033     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
03034     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
03035     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
03036     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
03037     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
03038     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
03039     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
03040     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
```

```

03041     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
03042     GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38>
03043 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
03044   T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
03045   T30, T31, T32, T33, T34, T35, T36, T37, T38, NoneT, NoneT, NoneT, NoneT,
03046   NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
03047   typedef Templates38<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
03048   T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
03049   T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38> type;
03050 };
03051 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
03052   GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
03053   GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
03054   GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
03055   GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
03056   GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
03057   GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
03058   GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
03059   GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
03060   GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
03061   GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
03062   GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
03063   GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39>
03064 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
03065   T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
03066   T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, NoneT, NoneT, NoneT,
03067   NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
03068   typedef Templates39<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
03069   T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
03070   T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39> type;
03071 };
03072 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
03073   GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
03074   GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
03075   GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
03076   GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
03077   GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
03078   GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
03079   GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
03080   GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
03081   GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
03082   GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
03083   GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
03084   GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
03085   GTEST_TEMPLATE_ T40>
03086 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
03087   T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
03088   T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, NoneT, NoneT, NoneT,
03089   NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
03090   typedef Templates40<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
03091   T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
03092   T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40> type;
03093 };
03094 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
03095   GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
03096   GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
03097   GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
03098   GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
03099   GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
03100   GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
03101   GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
03102   GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
03103   GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
03104   GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
03105   GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
03106   GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
03107   GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41>
03108 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
03109   T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
03110   T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, NoneT, NoneT,
03111   NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
03112   typedef Templates41<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
03113   T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
03114   T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40,
03115   T41> type;
03116 };
03117 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
03118   GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
03119   GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
03120   GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
03121   GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
03122   GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
03123   GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
03124   GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
03125   GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
03126   GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
03127   GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,

```

```
03128     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
03129     GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
03130     GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42>
03131 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
03132     T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
03133     T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, NoneT,
03134     NoneT, NoneT, NoneT, NoneT, NoneT> {
03135     typedef Templates42<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
03136     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
03137     T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41,
03138     T42> type;
03139 };
03140 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
03141     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
03142     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
03143     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
03144     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
03145     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
03146     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
03147     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
03148     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
03149     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
03150     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
03151     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
03152     GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
03153     GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
03154     GTEST_TEMPLATE_ T43>
03155     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
03156     T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
03157     T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
03158     NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
03159     typedef Templates43<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
03160     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
03161     T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41,
03162     T42, T43> type;
03163 };
03164 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
03165     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
03166     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
03167     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
03168     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
03169     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
03170     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
03171     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
03172     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
03173     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
03174     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
03175     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
03176     GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
03177     GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
03178     GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44>
03179     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
03180     T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
03181     T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44,
03182     NoneT, NoneT, NoneT, NoneT, NoneT> {
03183     typedef Templates44<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
03184     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
03185     T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41,
03186     T42, T43, T44> type;
03187 };
03188 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
03189     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
03190     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
03191     GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
03192     GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
03193     GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
03194     GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
03195     GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
03196     GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
03197     GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
03198     GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
03199     GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
03200     GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
03201     GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
03202     GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45>
03203     struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
03204     T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
03205     T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44,
03206     T45, NoneT, NoneT, NoneT, NoneT> {
03207     typedef Templates45<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
03208     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
03209     T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41,
03210     T42, T43, T44, T45> type;
03211 };
03212 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
03213     GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
03214     GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
```

```

03215 GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
03216 GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
03217 GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
03218 GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
03219 GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
03220 GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
03221 GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
03222 GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
03223 GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
03224 GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
03225 GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
03226 GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
03227 GTEST_TEMPLATE_ T46>
03228 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
03229 T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
03230 T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44,
03231 T45, T46, NoneT, NoneT, NoneT> {
03232 typedef Templates46<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
03233 T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
03234 T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41,
03235 T42, T43, T44, T45, T46> type;
03236 };
03237 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
03238 GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
03239 GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
03240 GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
03241 GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
03242 GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
03243 GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
03244 GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
03245 GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
03246 GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
03247 GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
03248 GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
03249 GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
03250 GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
03251 GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
03252 GTEST_TEMPLATE_ T46, GTEST_TEMPLATE_ T47>
03253 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
03254 T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
03255 T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44,
03256 T45, T46, T47, NoneT, NoneT, NoneT> {
03257 typedef Templates47<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
03258 T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
03259 T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41,
03260 T42, T43, T44, T45, T46, T47> type;
03261 };
03262 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
03263 GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
03264 GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
03265 GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
03266 GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
03267 GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
03268 GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
03269 GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
03270 GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
03271 GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
03272 GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
03273 GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
03274 GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
03275 GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
03276 GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
03277 GTEST_TEMPLATE_ T46, GTEST_TEMPLATE_ T47, GTEST_TEMPLATE_ T48>
03278 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
03279 T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
03280 T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44,
03281 T45, T46, T47, T48, NoneT, NoneT> {
03282 typedef Templates48<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
03283 T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
03284 T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41,
03285 T42, T43, T44, T45, T46, T47, T48> type;
03286 };
03287 template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
03288 GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
03289 GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
03290 GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
03291 GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
03292 GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
03293 GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
03294 GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
03295 GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
03296 GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
03297 GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
03298 GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
03299 GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
03300 GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
03301 GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,

```

```

03302     GTEST_TEMPLATE_ T46, GTEST_TEMPLATE_ T47, GTEST_TEMPLATE_ T48,
03303     GTEST_TEMPLATE_ T49>
03304 struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
03305     T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29,
03306     T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43, T44,
03307     T45, T46, T47, T48, T49, NoneT> {
03308     typedef Templates49<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
03309     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27,
03310     T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41,
03311     T42, T43, T44, T45, T46, T47, T48, T49> type;
03312 };
03313
03314 // The TypeList template makes it possible to use either a single type
03315 // or a Types<...> list in TYPED_TEST_CASE() and
03316 // INSTANTIATE_TYPED_TEST_CASE_P().
03317
03318 template <typename T>
03319 struct TypeList {
03320     typedef Types1<T> type;
03321 };
03322
03323 template <typename T1, typename T2, typename T3, typename T4, typename T5,
03324     typename T6, typename T7, typename T8, typename T9, typename T10,
03325     typename T11, typename T12, typename T13, typename T14, typename T15,
03326     typename T16, typename T17, typename T18, typename T19, typename T20,
03327     typename T21, typename T22, typename T23, typename T24, typename T25,
03328     typename T26, typename T27, typename T28, typename T29, typename T30,
03329     typename T31, typename T32, typename T33, typename T34, typename T35,
03330     typename T36, typename T37, typename T38, typename T39, typename T40,
03331     typename T41, typename T42, typename T43, typename T44, typename T45,
03332     typename T46, typename T47, typename T48, typename T49, typename T50>
03333 struct TypeList<Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
03334     T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, T27, T28,
03335     T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T42, T43,
03336     T44, T45, T46, T47, T48, T49, T50> > {
03337     typedef typename Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
03338         T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26,
03339         T27, T28, T29, T30, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40,
03340         T41, T42, T43, T44, T45, T46, T47, T48, T49, T50>::type type;
03341 };
03342
03343 #endif // GTEST_HAS_TYPED_TEST || GTEST_HAS_TYPED_TEST_P
03344
03345 } // namespace internal
03346 } // namespace testing
03347
03348 #endif // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_TYPE_UTIL_H_

```

9.53 Dokumentacja pliku pch.cpp

```
#include "pch.h"
```

9.54 Dokumentacja pliku pch.h

```
#include "gtest/gtest.h"
```

9.55 pch.h

[Idź do dokumentacji tego pliku.](#)

```

00001 //
00002 // pch.h
00003 //
00004
00005 #pragma once
00006
00007 #include "gtest/gtest.h"

```

9.56 Dokumentacja pliku test.cpp

```
#include "gtest/gtest.h"
#include "MergeSort.h"
#include <vector>
```

Funkcje

- [TEST \(MergeSortTest, AlreadySorted\)](#)
- [TEST \(MergeSortTest, ReversedOrder\)](#)
- [TEST \(MergeSortTest, RandomOrder\)](#)
- [TEST \(MergeSortTest, OnlyNegative\)](#)
- [TEST \(MergeSortTest, NegativeAndPositive\)](#)
- [TEST \(MergeSortTest, EmptyArray\)](#)
- [TEST \(MergeSortTest, SingleElement\)](#)
- [TEST \(MergeSortTest, WithDuplicates\)](#)
- [TEST \(MergeSortTest, NegativeDuplicates\)](#)
- [TEST \(MergeSortTest, MixedDuplicates\)](#)
- [TEST \(MergeSortTest, TwoElementsSorted\)](#)
- [TEST \(MergeSortTest, LargeArray\)](#)
- [TEST \(MergeSortTest, LargeArrayMixed\)](#)

9.56.1 Dokumentacja funkcji

9.56.1.1 TEST() [1/13]

```
TEST (
    MergeSortTest ,
    AlreadySorted )
```

9.56.1.2 TEST() [2/13]

```
TEST (
    MergeSortTest ,
    EmptyArray )
```

9.56.1.3 TEST() [3/13]

```
TEST (
    MergeSortTest ,
    LargeArray )
```

9.56.1.4 TEST() [4/13]

```
TEST (
    MergeSortTest ,
    LargeArrayMixed )
```

9.56.1.5 TEST() [5/13]

```
TEST (
    MergeSortTest ,
    MixedDuplicates )
```

9.56.1.6 TEST() [6/13]

```
TEST (
    MergeSortTest ,
    NegativeAndPositive )
```

9.56.1.7 TEST() [7/13]

```
TEST (
    MergeSortTest ,
    NegativeDuplicates )
```

9.56.1.8 TEST() [8/13]

```
TEST (
    MergeSortTest ,
    OnlyNegative )
```

9.56.1.9 TEST() [9/13]

```
TEST (
    MergeSortTest ,
    RandomOrder )
```

9.56.1.10 TEST() [10/13]

```
TEST (
    MergeSortTest ,
    ReversedOrder )
```

9.56.1.11 TEST() [11/13]

```
TEST (
    MergeSortTest ,
    SingleElement )
```

9.56.1.12 TEST() [12/13]

```
TEST (
    MergeSortTest ,
    TwoElementsSorted )
```

9.56.1.13 TEST() [13/13]

```
TEST (
    MergeSortTest ,
    WithDuplicates )
```

