MergeSortProject

Wygenerowano za pomocą Doxygen 1.15.0

Rozdział 1

Struktura katalogów

1.1 Katalogi

Rozdział 2

Indeks przestrzeni nazw

2.1 Lista przestrzeni nazw

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

Rozdział 3

Indeks hierarchiczny

3.1 Hierarchia klas

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

Rozdział 4

Indeks klas

4.1 Lista klas

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

Rozdział 5

Indeks plików

5.1 Lista plików

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

Rozdział 6

Dokumentacja katalogów

6.1 Dokumentacja katalogu packages/Microsoft.googletest.v140.↩ 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_(mut
- 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.googletest.v140.↩ windesktop.msvcstl.static.rt-dyn.1.8.1.8

Katalogi

- katalog build

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

Katalogi

- katalog include

6.9 Dokumentacja katalogu packages

Katalogi

- katalog Microsoft.googletest.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)

- 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 > Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, 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, T12 > 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)
- 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 > 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 > 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)
- 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 > 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)
- 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 > 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)
- 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 > 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)
- 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 > 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)
- 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 > 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)
- 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 > 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)
- 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 > 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)
- 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 > 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)

- 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 > 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)
- 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 > 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)
- 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 > 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)
- 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 > 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)
- 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 > 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)
- 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 > 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)
- 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 > 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)
- 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, ty-

pename 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)

- 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 > 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)
- 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 > 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)
- 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 > 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)
- 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 > 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)
- 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 > 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)
- 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 > 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)

- 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::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 > 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)
- 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>
  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 > 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)
- 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>
  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 > 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)
- 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 > 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)
- 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 > 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)

- 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 > 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)
- 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 > 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 > 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 > 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 > 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 > 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::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 > 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>
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, 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 ∗ TypeId
- 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

Wyliczenia

- 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_↩ 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_↩ 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 &gtest_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>
  TypeId GetTypeId ()
- GTEST_API_ TypeId GetTestTypeId ()
- GTEST_API_ TestInfo ∗ 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)
- 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 > GetArgvs ()
- GTEST_API_ size_t GetThreadCount ()
- bool IsAlpha (char ch)
- bool IsAlNum (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 GetArgvs()

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

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 GetTestTypeId()

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

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 kMaxBiggestInt

```
const BiggestInt testing::internal::kMaxBiggestInt
```

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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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
```

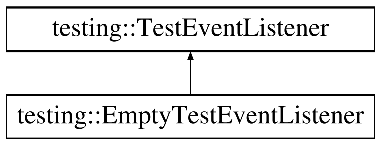
Dokumentacja dla tej struktury została wygenerowana z pliku:

- 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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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]
```

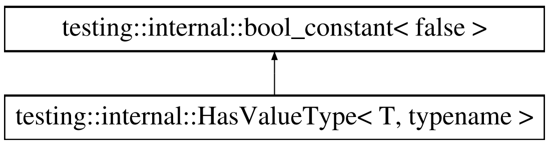
Dokumentacja dla tej klasy została wygenerowana z pliku:

- 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 >


testing::internal::bool_constant< false >
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

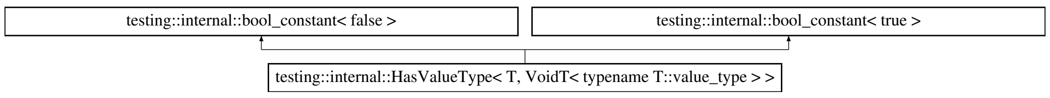
Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.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 > >


testing::internal::bool_constant< false >
testing::internal::bool_constant< true >
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.googletest.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
```

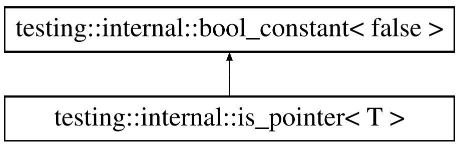
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.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 >


testing::internal::bool_constant< false >
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

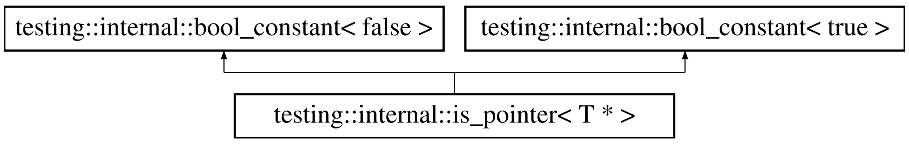
Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.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 ∗ >


testing::internal::bool_constant< false >
testing::internal::bool_constant< true >
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

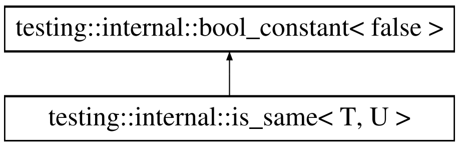
Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.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 >


testing::internal::bool_constant< false >
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

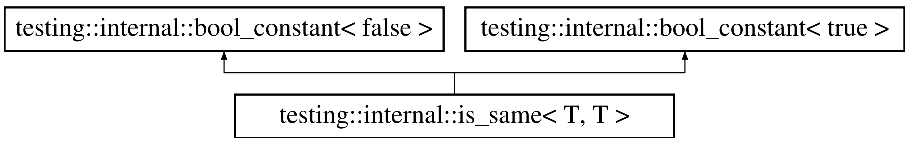
Dokumentacja dla tej struktury została wygenerowana z pliku:

- 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 >


testing::internal::bool_constant< false >
testing::internal::bool_constant< true >
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

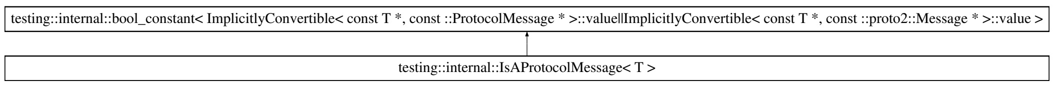
Dokumentacja dla tej struktury została wygenerowana z pliku:

- 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 >


testing::internal::bool_constant< ImplicitlyConvertible< const T *, const ::ProtocolMessage * >::value||ImplicitlyConvertible< const T *, const ::proto2::Message * >::value >
testing::internal::IsAProtocolMessage< T >

Dodatkowe dziedziczone składowe

Typy publiczne dziedziczone z testing::internal::bool_constant< ImplicitlyConvertible< const T ∗, const ::ProtocolMessage ∗ >::value

- typedef bool_constant< bool_value > type

Statyczne atrybuty publiczne dziedziczone z testing::internal::bool_constant< ImplicitlyConvertible< const T ∗, const ::ProtocolMessage ∗ >::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]
```

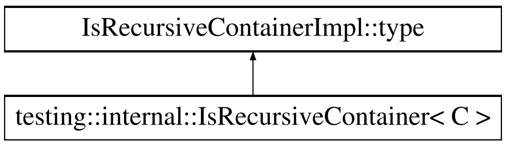
Dokumentacja dla tej struktury została wygenerowana z pliku:

- packages/Microsoft.googletest.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 >


IsRecursiveContainerImpl::type
testing::internal::IsRecursiveContainer< C >

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

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

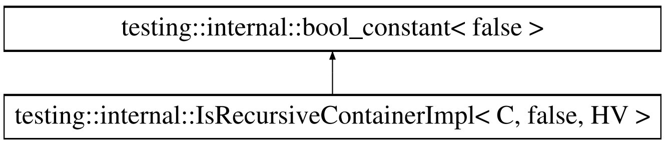
Dokumentacja dla tej struktury została wygenerowana z pliku:

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

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

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

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


testing::internal::bool_constant< false >
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

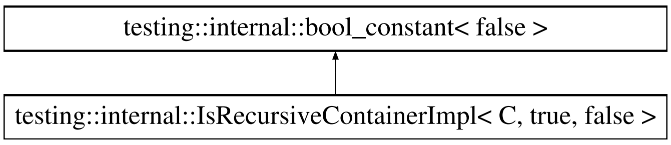
Dokumentacja dla tej struktury została wygenerowana z pliku:

- 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 >


testing::internal::bool_constant< false >
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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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]
```

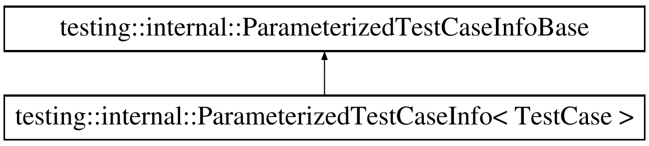
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.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 >


testing::internal::ParameterizedTestCaseInfoBase
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 TypeId GetTestCaseTypeId () 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 >::Param←
Type
```

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.

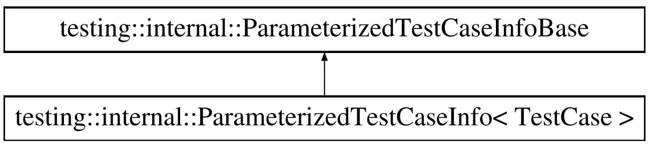
Dokumentacja dla tej klasy została wygenerowana z pliku:

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

8.64 Dokumentacja klasy testing::internal::ParameterizedTestCaseInfoBase

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

Diagram dziedziczenia dla testing::internal::ParameterizedTestCaseInfoBase


testing::internal::ParameterizedTestCaseInfoBase
testing::internal::ParameterizedTestCaseInfo< TestCase >

Metody publiczne

- virtual ∼ParameterizedTestCaseInfoBase ()
- virtual const std::string & GetTestCaseName () const =0
- virtual TypeId 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 TypeId 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.

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]
```

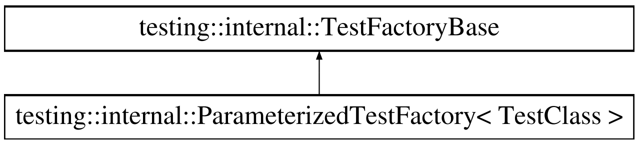
Dokumentacja dla tej klasy została wygenerowana z pliku:

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

8.66 Dokumentacja szablonu klasy testing::internal::ParameterizedTestFactory< TestClass >

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

Diagram dziedziczenia dla testing::internal::ParameterizedTestFactory< TestClass >


testing::internal::TestFactoryBase
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 >::Param←↩
Type
```

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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util.

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]
```

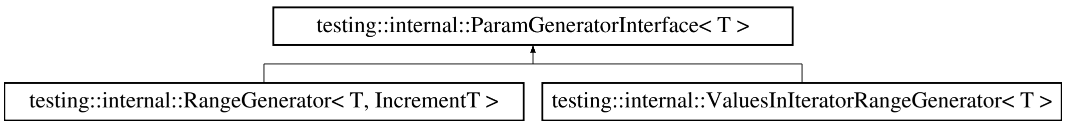
Dokumentacja dla tej klasy została wygenerowana z pliku:

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

8.68 Dokumentacja szablonu klasy testing::internal::ParamGeneratorInterface< T >

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

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


testing::internal::ParamGeneratorInterface< T >
testing::internal::RangeGenerator< T, IncrementT >
testing::internal::ValuesInIteratorRangeGenerator< 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::ValuesInIteratorRangeGenerator< 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::ValuesInIteratorRangeGenerator< 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.

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.

8.70 Dokumentacja szablonu klasy testing::internal::ParamIteratorInterface< T >

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

Metody publiczne

- virtual ∼ParamIteratorInterface ()
- virtual const ParamGeneratorInterface< 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.

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.

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.

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]
```

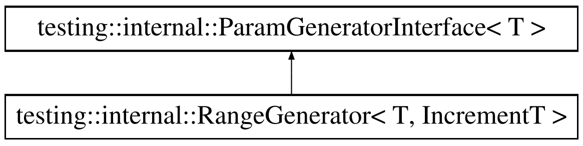
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.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 >


testing::internal::ParamGeneratorInterface< T >
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.

8.75 Dokumentacja klasy testing::internal::RE

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

Metody publiczne

- RE (const RE &other)
- RE (const ::std::string &regex)
- 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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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]
```

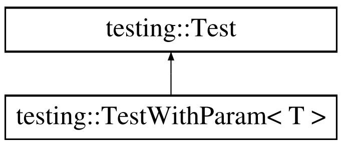
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.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]
```

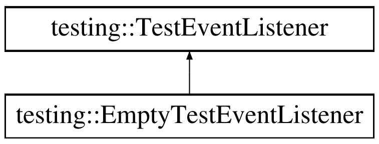
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.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.googletest.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]
```

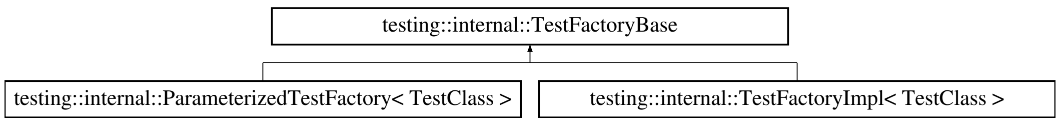
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.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


testing::internal::TestFactoryBase
testing::internal::ParameterizedTestFactory< TestClass >
testing::internal::TestFactoryImpl< TestClass >

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 >.

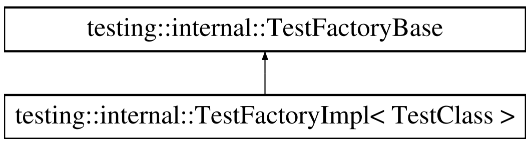
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.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 >


testing::internal::TestFactoryBase
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.googletest.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]
```

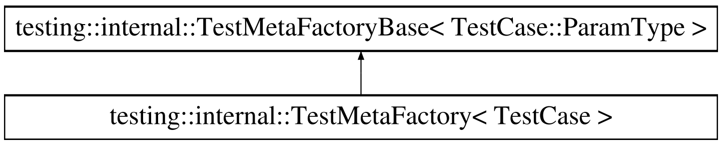
Dokumentacja dla tej klasy została wygenerowana z pliku:

- 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 >


testing::internal::TestMetaFactoryBase< TestCase::ParamType >
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 ∼TestMetaFactoryBase ()
- 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.

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 >::CreateTest←
Factory (
            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.

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.

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]
```

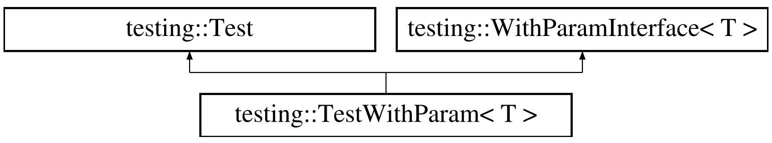
Dokumentacja dla tej klasy została wygenerowana z pliku:

- packages/Microsoft.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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::TypeIdHelper< 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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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.googletest.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-

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-

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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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-

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-

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>
testing::internal::ValueArray16< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, 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>
testing::internal::ValueArray16< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, 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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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>
testing::internal::ValueArray17< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
T15, T16, 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-

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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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-

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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

8.162 Dokumentacja szablonu klasy testing::internal::ValueArray20< 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-

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-

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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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-

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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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-

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-

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-

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-

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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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 >::Value↩
Array31 (
             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)  [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-

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-

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-

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,
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-

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>
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-

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
```

```
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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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
```

```
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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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>
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 (
             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
```

```
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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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-

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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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, 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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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 Dokumentacja konstruktora i destruktora

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>
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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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-

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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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>
```

Metody publiczne

- 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-

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 >

```
#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>
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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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 Dokumentacja funkcji składowych

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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util

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-

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-

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-

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 >::Value↩
Array50 (
             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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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-

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.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/internal/gtest-param-util-

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]
```

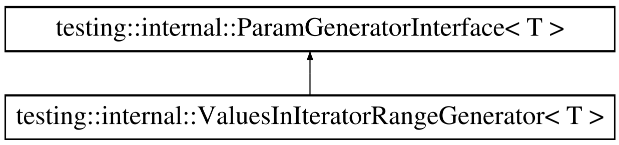
Dokumentacja dla tej klasy została wygenerowana z pliku:

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

8.200 Dokumentacja szablonu klasy testing::internal::ValuesInIteratorRangeGenerator< T >

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

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


testing::internal::ParamGeneratorInterface< T >
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.

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
```

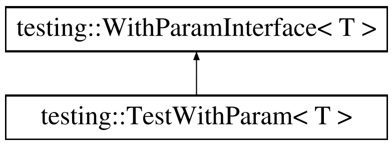
Dokumentacja dla tej struktury została wygenerowana z pliku:

- 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.googletest.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.googletest.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.

```

#ifndef MERGESORT_H
#define MERGESORT_H

#include <vector>

template<typename T>
class MergeSort {
public:
    static void sort(std::vector<T>& arr) {
        if (arr.size() <= 1) return;
        mergeSort(arr, 0, static_cast<int>(arr.size()) - 1);
    }

private:
    static void mergeSort(std::vector<T>& arr, int left, int right) {
        if (left >= right) return;

        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }

    static void merge(std::vector<T>& arr, int left, int mid, int right) {
        std::vector<T> temp(right - left + 1);
        int i = left, j = mid + 1, k = 0;

        while (i <= mid && j <= right)
            temp[k++] = (arr[i] < arr[j]) ? arr[i++] : arr[j++];

        while (i <= mid) temp[k++] = arr[i++];
        while (j <= right) temp[k++] = arr[j++];

        for (int x = 0; x < k; x++)
            arr[left + x] = temp[x];
    }
};
#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.

```
// Copyright 2005, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

//
// The Google C++ Testing and Mocking Framework (Google Test)
//
// This header file defines the public API for death tests.  It is
// #included by gtest.h so a user doesn't need to include this
// directly.
// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_GTEST_DEATH_TEST_H_
#define GTEST_INCLUDE_GTEST_GTEST_DEATH_TEST_H_

#include "gtest/internal/gtest-death-test-internal.h"

namespace testing {

// This flag controls the style of death tests.  Valid values are "threadsafe",
// meaning that the death test child process will re-execute the test binary
// from the start, running only a single death test, or "fast",
// meaning that the child process will execute the test logic immediately
// after forking.
GTEST_DECLARE_string_(death_test_style);

#if GTEST_HAS_DEATH_TEST

namespace internal {

// Returns a Boolean value indicating whether the caller is currently
// executing in the context of the death test child process.  Tools such as
// Valgrind heap checkers may need this to modify their behavior in death
// tests.  IMPORTANT: This is an internal utility.  Using it may break the
// implementation of death tests.  User code MUST NOT use it.
GTEST_API_ bool InDeathTestChild();

}  // namespace internal

// The following macros are useful for writing death tests.

// Here's what happens when an ASSERT_DEATH* or EXPECT_DEATH* is
// executed:
//
//   1. It generates a warning if there is more than one active
//   thread.  This is because it's safe to fork() or clone() only
//   when there is a single thread.
//
//   2. The parent process clone()s a sub-process and runs the death
//   test in it; the sub-process exits with code 0 at the end of the
//   death test, if it hasn't exited already.
//
//   3. The parent process waits for the sub-process to terminate.
//
//   4. The parent process checks the exit code and error message of
//   the sub-process.
//
```

```
// Examples:
//
//   ASSERT_DEATH(server.SendMessage(56, "Hello"), "Invalid port number");
//   for (int i = 0; i < 5; i++) {
//     EXPECT_DEATH(server.ProcessRequest(i),
//                  "Invalid request .* in ProcessRequest()")
//                  « "Failed to die on request " « i;
//   }
//
//   ASSERT_EXIT(server.ExitNow(), ::testing::ExitedWithCode(0), "Exiting");
//
//   bool KilledBySIGHUP(int exit_code) {
//     return WIFSIGNALED(exit_code) && WTERMSIG(exit_code) == SIGHUP;
//   }
//
//   ASSERT_EXIT(client.HangUpServer(), KilledBySIGHUP, "Hanging up!");
//
// On the regular expressions used in death tests:
//
//   GOOGLETEST_CM0005 DO NOT DELETE
//   On POSIX-compliant systems (*nix), we use the <regex.h> library,
//   which uses the POSIX extended regex syntax.
//
//   On other platforms (e.g. Windows or Mac), we only support a simple regex
//   syntax implemented as part of Google Test.  This limited
//   implementation should be enough most of the time when writing
//   death tests; though it lacks many features you can find in PCRE
//   or POSIX extended regex syntax.  For example, we don't support
//   union ("x|y"), grouping ("(xy)"), brackets ("[xy]"), and
//   repetition count ("x{5,7}"), among others.
//
//   Below is the syntax that we do support.  We chose it to be a
//   subset of both PCRE and POSIX extended regex, so it's easy to
//   learn wherever you come from.  In the following: 'A' denotes a
//   literal character, period (.), or a single \\ escape sequence;
//   'x' and 'y' denote regular expressions; 'm' and 'n' are for
//   natural numbers.
//
//     c     matches any literal character c
//     \\d   matches any decimal digit
//     \\D   matches any character that's not a decimal digit
//     \\f   matches \f
//     \\n   matches \n
//     \\r   matches \r
//     \\s   matches any ASCII whitespace, including \n
//     \\S   matches any character that's not a whitespace
//     \\t   matches \t
//     \\v   matches \v
//     \\w   matches any letter, _, or decimal digit
//     \\W   matches any character that \\w doesn't match
//     \\c   matches any literal character c, which must be a punctuation
//     .     matches any single character except \n
//     A?    matches 0 or 1 occurrences of A
//     A*    matches 0 or many occurrences of A
//     A+    matches 1 or many occurrences of A
//     ^     matches the beginning of a string (not that of each line)
//     $     matches the end of a string (not that of each line)
//     xy    matches x followed by y
//
//   If you accidentally use PCRE or POSIX extended regex features
//   not implemented by us, you will get a run-time failure.  In that
//   case, please try to rewrite your regular expression within the
//   above syntax.
//
//   This implementation is *not* meant to be as highly tuned or robust
//   as a compiled regex library, but should perform well enough for a
//   death test, which already incurs significant overhead by launching
//   a child process.
//
// Known caveats:
//
//   A "threadsafe" style death test obtains the path to the test
//   program from argv[0] and re-executes it in the sub-process.  For
//   simplicity, the current implementation doesn't search the PATH
//   when launching the sub-process.  This means that the user must
//   invoke the test program via a path that contains at least one
//   path separator (e.g. path/to/foo_test and
//   /absolute/path/to/bar_test are fine, but foo_test is not).  This
//   is rarely a problem as people usually don't put the test binary
//   directory in PATH.
//
// FIXME: make thread-safe death tests search the PATH.

// Asserts that a given statement causes the program to exit, with an
// integer exit status that satisfies predicate, and emitting error output
// that matches regex.
# define ASSERT_EXIT(statement, predicate, regex) \
    GTEST_DEATH_TEST_(statement, predicate, regex, GTEST_FATAL_FAILURE_)

// Like ASSERT_EXIT, but continues on to successive tests in the
// test case, if any:
# define EXPECT_EXIT(statement, predicate, regex) \
    GTEST_DEATH_TEST_(statement, predicate, regex, GTEST_NONFATAL_FAILURE_)

// Asserts that a given statement causes the program to exit, either by
// explicitly exiting with a nonzero exit code or being killed by a
// signal, and emitting error output that matches regex.
# define ASSERT_DEATH(statement, regex) \
    ASSERT_EXIT(statement, ::testing::internal::ExitedUnsuccessfully, regex)

// Like ASSERT_DEATH, but continues on to successive tests in the
// test case, if any:
# define EXPECT_DEATH(statement, regex) \
    EXPECT_EXIT(statement, ::testing::internal::ExitedUnsuccessfully, regex)

// Two predicate classes that can be used in {ASSERT,EXPECT}_EXIT*:

// Tests that an exit code describes a normal exit with a given exit code.
class GTEST_API_ ExitedWithCode {
 public:
  explicit ExitedWithCode(int exit_code);
  bool operator()(int exit_status) const;
 private:
  // No implementation - assignment is unsupported.
  void operator=(const ExitedWithCode& other);

  const int exit_code_;
};

# if !GTEST_OS_WINDOWS && !GTEST_OS_FUCHSIA
// Tests that an exit code describes an exit due to termination by a
// given signal.
// GOOGLETEST_CM0006 DO NOT DELETE
class GTEST_API_ KilledBySignal {
 public:
  explicit KilledBySignal(int signum);
  bool operator()(int exit_status) const;
 private:
  const int signum_;
};
# endif  // !GTEST_OS_WINDOWS

// EXPECT_DEBUG_DEATH asserts that the given statements die in debug mode.
// The death testing framework causes this to have interesting semantics,
// since the sideeffects of the call are only visible in opt mode, and not
// in debug mode.
//
// In practice, this can be used to test functions that utilize the
// LOG(DFATAL) macro using the following style:
//
// int DieInDebugOr12(int* sideeffect) {
//   if (sideeffect) {
//     *sideeffect = 12;
//   }
//   LOG(DFATAL) « "death";
//   return 12;
// }
//
// TEST(TestCase, TestDieOr12WorksInDgbAndOpt) {
//   int sideeffect = 0;
//   // Only asserts in dbg.
//   EXPECT_DEBUG_DEATH(DieInDebugOr12(&sideeffect), "death");
//
// #ifdef NDEBUG
//   // opt-mode has sideeffect visible.
//   EXPECT_EQ(12, sideeffect);
// #else
//   // dbg-mode no visible sideeffect.
//   EXPECT_EQ(0, sideeffect);
// #endif
// }
//
// This will assert that DieInDebugReturn12InOpt() crashes in debug
// mode, usually due to a DCHECK or LOG(DFATAL), but returns the
// appropriate fallback value (12 in this case) in opt mode. If you
// need to test that a function has appropriate side-effects in opt
// mode, include assertions against the side-effects.  A general
// pattern for this is:
//
// EXPECT_DEBUG_DEATH({
//   // Side-effects here will have an effect after this statement in
//   // opt mode, but none in debug mode.
//   EXPECT_EQ(12, DieInDebugOr12(&sideeffect));
// }, "death");
```

```
//
# ifdef NDEBUG

#  define EXPECT_DEBUG_DEATH(statement, regex) \
  GTEST_EXECUTE_STATEMENT_(statement, regex)

#  define ASSERT_DEBUG_DEATH(statement, regex) \
  GTEST_EXECUTE_STATEMENT_(statement, regex)

# else

#  define EXPECT_DEBUG_DEATH(statement, regex) \
  EXPECT_DEATH(statement, regex)

#  define ASSERT_DEBUG_DEATH(statement, regex) \
  ASSERT_DEATH(statement, regex)

# endif  // NDEBUG for EXPECT_DEBUG_DEATH
#endif  // GTEST_HAS_DEATH_TEST

// This macro is used for implementing macros such as
// EXPECT_DEATH_IF_SUPPORTED and ASSERT_DEATH_IF_SUPPORTED on systems where
// death tests are not supported. Those macros must compile on such systems
// iff EXPECT_DEATH and ASSERT_DEATH compile with the same parameters on
// systems that support death tests. This allows one to write such a macro
// on a system that does not support death tests and be sure that it will
// compile on a death-test supporting system. It is exposed publicly so that
// systems that have death-tests with stricter requirements than
// GTEST_HAS_DEATH_TEST can write their own equivalent of
// EXPECT_DEATH_IF_SUPPORTED and ASSERT_DEATH_IF_SUPPORTED.
//
// Parameters:
//   statement -  A statement that a macro such as EXPECT_DEATH would test
//                for program termination. This macro has to make sure this
//                statement is compiled but not executed, to ensure that
//                EXPECT_DEATH_IF_SUPPORTED compiles with a certain
//                parameter iff EXPECT_DEATH compiles with it.
//   regex     -  A regex that a macro such as EXPECT_DEATH would use to test
//                the output of statement.  This parameter has to be
//                compiled but not evaluated by this macro, to ensure that
//                this macro only accepts expressions that a macro such as
//                EXPECT_DEATH would accept.
//   terminator - Must be an empty statement for EXPECT_DEATH_IF_SUPPORTED
//                and a return statement for ASSERT_DEATH_IF_SUPPORTED.
//                This ensures that ASSERT_DEATH_IF_SUPPORTED will not
//                compile inside functions where ASSERT_DEATH doesn’t
//                compile.
//
//  The branch that has an always false condition is used to ensure that
//  statement and regex are compiled (and thus syntactically correct) but
//  never executed. The unreachable code macro protects the terminator
//  statement from generating an ’unreachable code’ warning in case
//  statement unconditionally returns or throws. The Message constructor at
//  the end allows the syntax of streaming additional messages into the
//  macro, for compilational compatibility with EXPECT_DEATH/ASSERT_DEATH.
# define GTEST_UNSUPPORTED_DEATH_TEST(statement, regex, terminator) \
    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()

// EXPECT_DEATH_IF_SUPPORTED(statement, regex) and
// ASSERT_DEATH_IF_SUPPORTED(statement, regex) expand to real death tests if
// death tests are supported; otherwise they just issue a warning.  This is
// useful when you are combining death test assertions with normal test
// assertions in one test.
#if GTEST_HAS_DEATH_TEST
# define EXPECT_DEATH_IF_SUPPORTED(statement, regex) \
    EXPECT_DEATH(statement, regex)
# define ASSERT_DEATH_IF_SUPPORTED(statement, regex) \
    ASSERT_DEATH(statement, regex)
#else
# define EXPECT_DEATH_IF_SUPPORTED(statement, regex) \
    GTEST_UNSUPPORTED_DEATH_TEST(statement, regex, )
# define ASSERT_DEATH_IF_SUPPORTED(statement, regex) \
    GTEST_UNSUPPORTED_DEATH_TEST(statement, regex, return)
#endif

}  // namespace testing

```

```
#endif  // GTEST_INCLUDE_GTEST_GTEST_DEATH_TEST_H_
```

9.6 Dokumentacja pliku packages/Microsoft.googletest.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.

```
// Copyright 2005, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

//
```

```
// The Google C++ Testing and Mocking Framework (Google Test)
//
// This header file defines the Message class.
//
// IMPORTANT NOTE: Due to limitation of the C++ language, we have to
// leave some internal implementation details in this header file.
// They are clearly marked by comments like this:
//
//   // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
//
// Such code is NOT meant to be used by a user directly, and is subject
// to CHANGE WITHOUT NOTICE.  Therefore DO NOT DEPEND ON IT in a user
// program!

// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_GTEST_MESSAGE_H_
#define GTEST_INCLUDE_GTEST_GTEST_MESSAGE_H_

#include <limits>

#include "gtest/internal/gtest-port.h"

GTEST_DISABLE_MSC_WARNINGS_PUSH_(4251 \
/* class A needs to have dll-interface to be used by clients of class B */)

// Ensures that there is at least one operator« in the global namespace.
// See Message& operator«(...) below for why.
void operator«(const testing::internal::Secret&, int);

namespace testing {

// The Message class works like an ostream repeater.
//
// Typical usage:
//
//   1. You stream a bunch of values to a Message object.
//      It will remember the text in a stringstream.
//   2. Then you stream the Message object to an ostream.
//      This causes the text in the Message to be streamed
//      to the ostream.
//
// For example;
//
//   testing::Message foo;
//   foo « 1 « " != " « 2;
//   std::cout « foo;
//
// will print "1 != 2".
//
// Message is not intended to be inherited from.  In particular, its
// destructor is not virtual.
//
// Note that stringstream behaves differently in gcc and in MSVC.  You
// can stream a NULL char pointer to it in the former, but not in the
// latter (it causes an access violation if you do).  The Message
// class hides this difference by treating a NULL char pointer as
// "(null)".
class GTEST_API_ Message {
 private:
  // The type of basic IO manipulators (endl, ends, and flush) for
  // narrow streams.
  typedef std::ostream& (*BasicNarrowIoManip)(std::ostream&);

 public:
  // Constructs an empty Message.
  Message();

  // Copy constructor.
  Message(const Message& msg) : ss_(new ::std::stringstream) {  // NOLINT
    *ss_ « msg.GetString();
  }

  // Constructs a Message from a C-string.
  explicit Message(const char* str) : ss_(new ::std::stringstream) {
    *ss_ « str;
  }

#if GTEST_OS_SYMBIAN
  // Streams a value (either a pointer or not) to this object.
  template <typename T>
  inline Message& operator «(const T& value) {
    StreamHelper(typename internal::is_pointer<T>::type(), value);
    return *this;
  }
#else
  // Streams a non-pointer value to this object.
  template <typename T>
  inline Message& operator «(const T& val) {
    // Some libraries overload « for STL containers.  These
    // overloads are defined in the global namespace instead of ::std.
    //
    // C++'s symbol lookup rule (i.e. Koenig lookup) says that these
    // overloads are visible in either the std namespace or the global
    // namespace, but not other namespaces, including the testing
    // namespace which Google Test's Message class is in.
    //
    // To allow STL containers (and other types that has a « operator
    // defined in the global namespace) to be used in Google Test
    // assertions, testing::Message must access the custom « operator
    // from the global namespace.  With this using declaration,
    // overloads of « defined in the global namespace and those
    // visible via Koenig lookup are both exposed in this function.
    using ::operator «;
    *ss_ « val;
    return *this;
  }

  // Streams a pointer value to this object.
  //
  // This function is an overload of the previous one.  When you
  // stream a pointer to a Message, this definition will be used as it
  // is more specialized.  (The C++ Standard, section
  // [temp.func.order].)  If you stream a non-pointer, then the
  // previous definition will be used.
  //
  // The reason for this overload is that streaming a NULL pointer to
  // ostream is undefined behavior.  Depending on the compiler, you
  // may get "0", "(nil)", "(null)", or an access violation.  To
  // ensure consistent result across compilers, we always treat NULL
  // as "(null)".
  template <typename T>
  inline Message& operator «(T* const& pointer) {  // NOLINT
    if (pointer == NULL) {
      *ss_ « "(null)";
    } else {
      *ss_ « pointer;
    }
    return *this;
  }
#endif  // GTEST_OS_SYMBIAN

  // Since the basic IO manipulators are overloaded for both narrow
  // and wide streams, we have to provide this specialized definition
  // of operator «, even though its body is the same as the
  // templatized version above.  Without this definition, streaming
  // endl or other basic IO manipulators to Message will confuse the
  // compiler.
  Message& operator «(BasicNarrowIoManip val) {
    *ss_ « val;
    return *this;
  }

  // Instead of 1/0, we want to see true/false for bool values.
  Message& operator «(bool b) {
    return *this « (b ? "true" : "false");
  }

  // These two overloads allow streaming a wide C string to a Message
  // using the UTF-8 encoding.
  Message& operator «(const wchar_t* wide_c_str);
  Message& operator «(wchar_t* wide_c_str);

#if GTEST_HAS_STD_WSTRING
  // Converts the given wide string to a narrow string using the UTF-8
  // encoding, and streams the result to this Message object.
  Message& operator «(const ::std::wstring& wstr);
#endif  // GTEST_HAS_STD_WSTRING

#if GTEST_HAS_GLOBAL_WSTRING
  // Converts the given wide string to a narrow string using the UTF-8
  // encoding, and streams the result to this Message object.
  Message& operator «(const ::wstring& wstr);
#endif  // GTEST_HAS_GLOBAL_WSTRING

  // Gets the text streamed to this object so far as an std::string.
  // Each '\0' character in the buffer is replaced with "\\0".
  //
  // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
  std::string GetString() const;

 private:
#if GTEST_OS_SYMBIAN
  // These are needed as the Nokia Symbian Compiler cannot decide between
  // const T& and const T* in a function template. The Nokia compiler _can_
  // decide between class template specializations for T and T*, so a
  // tr1::type_traits-like is_pointer works, and we can overload on that.
  template <typename T>
  inline void StreamHelper(internal::true_type /*is_pointer*/, T* pointer) {
    if (pointer == NULL) {
      *ss_ « "(null)";
    } else {
      *ss_ « pointer;
    }
  }
  template <typename T>
  inline void StreamHelper(internal::false_type /*is_pointer*/,
                           const T& value) {
    // See the comments in Message& operator «(const T&) above for why
    // we need this using statement.
    using ::operator «;
    *ss_ « value;
  }
#endif  // GTEST_OS_SYMBIAN

  // We'll hold the text streamed to this object here.
  const internal::scoped_ptr< ::std::stringstream> ss_;

  // We declare (but don't implement) this to prevent the compiler
  // from implementing the assignment operator.
  void operator=(const Message&);
};

// Streams a Message to an ostream.
inline std::ostream& operator «(std::ostream& os, const Message& sb) {
  return os « sb.GetString();
}

namespace internal {

// Converts a streamable value to an std::string.  A NULL pointer is
// converted to "(null)".  When the input value is a ::string,
// ::std::string, ::wstring, or ::std::wstring object, each NUL
// character in it is replaced with "\\0".
template <typename T>
std::string StreamableToString(const T& streamable) {
  return (Message() « streamable).GetString();
}

}  // namespace internal
}  // namespace testing

GTEST_DISABLE_MSC_WARNINGS_POP_()  //  4251

#endif  // GTEST_INCLUDE_GTEST_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>
  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 > testing::Values (T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9, T10 v10, 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, 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)
- 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)
- 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)

- 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)
- 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)
- 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)
- 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)
- 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)
- 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)
- 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)
- 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)
- 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)
- 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)

- 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)
- 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)
- 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)
- 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)
- 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)
- 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)
- 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)

- 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)
- 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)
- 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)
- 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)
- 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)
- 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::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 > 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)

- 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>
  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)
- 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>
  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)
- 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)
- 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)
- 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)
- 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, type-

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.

```
// This file was GENERATED by command:
//     pump.py gtest-param-test.h.pump
// DO NOT EDIT BY HAND!!!

// Copyright 2008, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// Macros and functions for implementing parameterized tests
// in Google C++ Testing and Mocking Framework (Google Test)
//
// This file is generated by a SCRIPT.  DO NOT EDIT BY HAND!
//
// GOOGLETEST_CM0001 DO NOT DELETE
#ifndef GTEST_INCLUDE_GTEST_GTEST_PARAM_TEST_H_
#define GTEST_INCLUDE_GTEST_GTEST_PARAM_TEST_H_


// Value-parameterized tests allow you to test your code with different
// parameters without writing multiple copies of the same test.
//
// Here is how you use value-parameterized tests:

#if 0

// To write value-parameterized tests, first you should define a fixture
// class. It is usually derived from testing::TestWithParam<T> (see below for
// another inheritance scheme that's sometimes useful in more complicated
// class hierarchies), where the type of your parameter values.
// TestWithParam<T> is itself derived from testing::Test. T can be any
// copyable type. If it's a raw pointer, you are responsible for managing the
// lifespan of the pointed values.

class FooTest : public ::testing::TestWithParam<const char*> {
  // You can implement all the usual class fixture members here.
};

// Then, use the TEST_P macro to define as many parameterized tests
// for this fixture as you want. The _P suffix is for "parameterized"
// or "pattern", whichever you prefer to think.

TEST_P(FooTest, DoesBlah) {
  // Inside a test, access the test parameter with the GetParam() method
  // of the TestWithParam<T> class:
  EXPECT_TRUE(foo.Blah(GetParam()));
  ...
}

TEST_P(FooTest, HasBlahBlah) {
  ...
}

// Finally, you can use INSTANTIATE_TEST_CASE_P to instantiate the test
// case with any set of parameters you want. Google Test defines a number
// of functions for generating test parameters. They return what we call
// (surprise!) parameter generators. Here is a summary of them, which
// are all in the testing namespace:
//
//
//  Range(begin, end [, step]) - Yields values {begin, begin+step,
//                               begin+step+step, ...}. The values do not
//                               include end. step defaults to 1.
//  Values(v1, v2, ..., vN)    - Yields values {v1, v2, ..., vN}.
//  ValuesIn(container)        - Yields values from a C-style array, an STL
//  ValuesIn(begin,end)          container, or an iterator range [begin, end).
//  Bool()                     - Yields sequence {false, true}.
//  Combine(g1, g2, ..., gN)   - Yields all combinations (the Cartesian product
//                               for the math savvy) of the values generated
//                               by the N generators.
//
// For more details, see comments at the definitions of these functions below
// in this file.
//
// The following statement will instantiate tests from the FooTest test case
// each with parameter values "meeny", "miny", and "moe".

INSTANTIATE_TEST_CASE_P(InstantiationName,
                        FooTest,
                        Values("meeny", "miny", "moe"));

```

```
// To distinguish different instances of the pattern, (yes, you
// can instantiate it more then once) the first argument to the
// INSTANTIATE_TEST_CASE_P macro is a prefix that will be added to the
// actual test case name. Remember to pick unique prefixes for different
// instantiations. The tests from the instantiation above will have
// these names:
//
//    * InstantiationName/FooTest.DoesBlah/0 for "meeny"
//    * InstantiationName/FooTest.DoesBlah/1 for "miny"
//    * InstantiationName/FooTest.DoesBlah/2 for "moe"
//    * InstantiationName/FooTest.HasBlahBlah/0 for "meeny"
//    * InstantiationName/FooTest.HasBlahBlah/1 for "miny"
//    * InstantiationName/FooTest.HasBlahBlah/2 for "moe"
//
// You can use these names in --gtest_filter.
//
// This statement will instantiate all tests from FooTest again, each
// with parameter values "cat" and "dog":

const char* pets[] = {"cat", "dog"};
INSTANTIATE_TEST_CASE_P(AnotherInstantiationName, FooTest, ValuesIn(pets));

// The tests from the instantiation above will have these names:
//
//    * AnotherInstantiationName/FooTest.DoesBlah/0 for "cat"
//    * AnotherInstantiationName/FooTest.DoesBlah/1 for "dog"
//    * AnotherInstantiationName/FooTest.HasBlahBlah/0 for "cat"
//    * AnotherInstantiationName/FooTest.HasBlahBlah/1 for "dog"
//
// Please note that INSTANTIATE_TEST_CASE_P will instantiate all tests
// in the given test case, whether their definitions come before or
// AFTER the INSTANTIATE_TEST_CASE_P statement.
//
// Please also note that generator expressions (including parameters to the
// generators) are evaluated in InitGoogleTest(), after main() has started.
// This allows the user on one hand, to adjust generator parameters in order
// to dynamically determine a set of tests to run and on the other hand,
// give the user a chance to inspect the generated tests with Google Test
// reflection API before RUN_ALL_TESTS() is executed.
//
// You can see samples/sample7_unittest.cc and samples/sample8_unittest.cc
// for more examples.
//
// In the future, we plan to publish the API for defining new parameter
// generators. But for now this interface remains part of the internal
// implementation and is subject to change.
//
//
// A parameterized test fixture must be derived from testing::Test and from
// testing::WithParamInterface<T>, where T is the type of the parameter
// values. Inheriting from TestWithParam<T> satisfies that requirement because
// TestWithParam<T> inherits from both Test and WithParamInterface. In more
// complicated hierarchies, however, it is occasionally useful to inherit
// separately from Test and WithParamInterface. For example:

class BaseTest : public ::testing::Test {
  // You can inherit all the usual members for a non-parameterized test
  // fixture here.
};

class DerivedTest : public BaseTest, public ::testing::WithParamInterface<int> {
  // The usual test fixture members go here too.
};

TEST_F(BaseTest, HasFoo) {
  // This is an ordinary non-parameterized test.
}

TEST_P(DerivedTest, DoesBlah) {
  // GetParam works just the same here as if you inherit from TestWithParam.
  EXPECT_TRUE(foo.Blah(GetParam()));
}

#endif  // 0

#include "gtest/internal/gtest-port.h"

#if !GTEST_OS_SYMBIAN
# include <utility>
#endif

#include "gtest/internal/gtest-internal.h"
#include "gtest/internal/gtest-param-util.h"
#include "gtest/internal/gtest-param-util-generated.h"

namespace testing {

```

```
// Functions producing parameter generators.
//
// Google Test uses these generators to produce parameters for value-
// parameterized tests. When a parameterized test case is instantiated
// with a particular generator, Google Test creates and runs tests
// for each element in the sequence produced by the generator.
//
// In the following sample, tests from test case FooTest are instantiated
// each three times with parameter values 3, 5, and 8:
//
// class FooTest : public TestWithParam<int> { ... };
//
// TEST_P(FooTest, TestThis) {
// }
// TEST_P(FooTest, TestThat) {
// }
// INSTANTIATE_TEST_CASE_P(TestSequence, FooTest, Values(3, 5, 8));
//

// Range() returns generators providing sequences of values in a range.
//
// Synopsis:
// Range(start, end)
//   - returns a generator producing a sequence of values {start, start+1,
//     start+2, ..., }.
// Range(start, end, step)
//   - returns a generator producing a sequence of values {start, start+step,
//     start+step+step, ..., }.
// Notes:
//   * The generated sequences never include end. For example, Range(1, 5)
//     returns a generator producing a sequence {1, 2, 3, 4}. Range(1, 9, 2)
//     returns a generator producing {1, 3, 5, 7}.
//   * start and end must have the same type. That type may be any integral or
//     floating-point type or a user defined type satisfying these conditions:
//     * It must be assignable (have operator=() defined).
//     * It must have operator+() (operator+(int-compatible type) for
//       two-operand version).
//     * It must have operator<() defined.
//     Elements in the resulting sequences will also have that type.
//   * Condition start < end must be satisfied in order for resulting sequences
//     to contain any elements.
//
template <typename T, typename IncrementT>
internal::ParamGenerator<T> Range(T start, T end, IncrementT step) {
  return internal::ParamGenerator<T>(
      new internal::RangeGenerator<T, IncrementT>(start, end, step));
}

template <typename T>
internal::ParamGenerator<T> Range(T start, T end) {
  return Range(start, end, 1);
}

// ValuesIn() function allows generation of tests with parameters coming from
// a container.
//
// Synopsis:
// ValuesIn(const T (&array)[N])
//   - returns a generator producing sequences with elements from
//     a C-style array.
// ValuesIn(const Container& container)
//   - returns a generator producing sequences with elements from
//     an STL-style container.
// ValuesIn(Iterator begin, Iterator end)
//   - returns a generator producing sequences with elements from
//     a range [begin, end) defined by a pair of STL-style iterators. These
//     iterators can also be plain C pointers.
//
// Please note that ValuesIn copies the values from the containers
// passed in and keeps them to generate tests in RUN_ALL_TESTS().
//
// Examples:
//
// This instantiates tests from test case StringTest
// each with C-string values of "foo", "bar", and "baz":
//
// const char* strings[] = {"foo", "bar", "baz"};
// INSTANTIATE_TEST_CASE_P(StringSequence, StringTest, ValuesIn(strings));
//
// This instantiates tests from test case StlStringTest
// each with STL strings with values "a" and "b":
//
// ::std::vector< ::std::string> GetParameterStrings() {
//   ::std::vector< ::std::string> v;
//   v.push_back("a");
//   v.push_back("b");
//   return v;
// }
//
// INSTANTIATE_TEST_CASE_P(CharSequence,
//                         StlStringTest,
//                         ValuesIn(GetParameterStrings()));
//
//
// This will also instantiate tests from CharTest
// each with parameter values 'a' and 'b':
//
// ::std::list<char> GetParameterChars() {
//   ::std::list<char> list;
//   list.push_back('a');
//   list.push_back('b');
//   return list;
// }
// ::std::list<char> l = GetParameterChars();
// INSTANTIATE_TEST_CASE_P(CharSequence2,
//                         CharTest,
//                         ValuesIn(l.begin(), l.end()));
//
template <typename ForwardIterator>
internal::ParamGenerator<
  typename ::testing::internal::IteratorTraits<ForwardIterator>::value_type>
ValuesIn(ForwardIterator begin, ForwardIterator end) {
  typedef typename ::testing::internal::IteratorTraits<ForwardIterator>
      ::value_type ParamType;
  return internal::ParamGenerator<ParamType>(
      new internal::ValuesInIteratorRangeGenerator<ParamType>(begin, end));
}

template <typename T, size_t N>
internal::ParamGenerator<T> ValuesIn(const T (&array)[N]) {
  return ValuesIn(array, array + N);
}

template <class Container>
internal::ParamGenerator<typename Container::value_type> ValuesIn(
    const Container& container) {
  return ValuesIn(container.begin(), container.end());
}

// Values() allows generating tests from explicitly specified list of
// parameters.
//
// Synopsis:
// Values(T v1, T v2, ..., T vN)
//   - returns a generator producing sequences with elements v1, v2, ..., vN.
//
// For example, this instantiates tests from test case BarTest each
// with values "one", "two", and "three":
//
// INSTANTIATE_TEST_CASE_P(NumSequence, BarTest, Values("one", "two", "three"));
//
// This instantiates tests from test case BazTest each with values 1, 2, 3.5.
// The exact type of values will depend on the type of parameter in BazTest.
//
// INSTANTIATE_TEST_CASE_P(FloatingNumbers, BazTest, Values(1, 2, 3.5));
//
// Currently, Values() supports from 1 to 50 parameters.
//
template <typename T1>
internal::ValueArray1<T1> Values(T1 v1) {
  return internal::ValueArray1<T1>(v1);
}

template <typename T1, typename T2>
internal::ValueArray2<T1, T2> Values(T1 v1, T2 v2) {
  return internal::ValueArray2<T1, T2>(v1, v2);
}

template <typename T1, typename T2, typename T3>
internal::ValueArray3<T1, T2, T3> Values(T1 v1, T2 v2, T3 v3) {
  return internal::ValueArray3<T1, T2, T3>(v1, v2, 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) {
  return internal::ValueArray4<T1, T2, T3, T4>(v1, v2, v3, 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) {
  return internal::ValueArray5<T1, T2, T3, T4, T5>(v1, v2, v3, v4, 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) {
  return internal::ValueArray6<T1, T2, T3, T4, T5, T6>(v1, v2, v3, v4, v5, 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) {
  return internal::ValueArray7<T1, T2, T3, T4, T5, T6, T7>(v1, v2, v3, v4, v5,
      v6, 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) {
  return internal::ValueArray8<T1, T2, T3, T4, T5, T6, T7, T8>(v1, v2, v3, v4,
      v5, v6, v7, 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) {
  return internal::ValueArray9<T1, T2, T3, T4, T5, T6, T7, T8, T9>(v1, v2, v3,
      v4, v5, v6, v7, v8, 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) {
  return internal::ValueArray10<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10>(v1,
      v2, v3, v4, v5, v6, v7, v8, v9, 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> Values(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
    T10 v10, T11 v11) {
  return internal::ValueArray11<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10,
      T11>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, 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,
    T12> 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) {
  return internal::ValueArray12<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
      T12>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, 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, T12,
    T13> 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) {
  return internal::ValueArray13<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
      T12, T13>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, 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> 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) {
  return internal::ValueArray14<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
      T12, T13, T14>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13,
      v14);
}

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

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> 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) {
  return internal::ValueArray16<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
      T12, T13, T14, T15, T16>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11,
      v12, v13, v14, v15, v16);
}

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

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> 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) {
  return internal::ValueArray18<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
      T12, T13, T14, T15, T16, T17, T18>(v1, v2, v3, v4, v5, v6, v7, v8, v9,
      v10, v11, v12, v13, v14, v15, v16, v17, v18);
}

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

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> 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) {
  return internal::ValueArray20<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
      T12, T13, T14, T15, T16, T17, T18, T19, T20>(v1, v2, v3, v4, v5, v6, v7,
      v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20);
}

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

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

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> 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) {
  return 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>(v1, v2, v3,
      v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19,
      v20, v21, v22, v23);
}

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> 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) {
  return 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>(v1, v2,
      v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18,
      v19, v20, v21, v22, v23, v24);
}

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> 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) {
  return 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>(v1,
      v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17,
      v18, v19, v20, v21, v22, v23, v24, v25);
}

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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15,
      v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26);
}

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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14,
      v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26, v27);
}

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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13,
      v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26, v27,
      v28);
}

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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12,
      v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26,
      v27, v28, v29);
}

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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11,
      v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25,
      v26, v27, v28, v29, v30);
}

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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10,
      v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24,
      v25, v26, v27, v28, v29, v30, v31);
}

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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8, v9,
      v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23,
      v24, v25, v26, v27, v28, v29, v30, v31, v32);
}

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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8,
      v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23,
      v24, v25, v26, v27, v28, v29, v30, v31, v32, v33);
}

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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7,
      v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22,
      v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33, v34);
}

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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6,
      v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21,
      v22, v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33, v34, v35);
}

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> 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) {
  return 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>(v1, v2, v3, v4,
      v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19,
      v20, v21, v22, v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33,
      v34, v35, v36);
}

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::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> 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) {
  return 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>(v1, v2, v3,
      v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19,
      v20, v21, v22, v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33,
      v34, v35, v36, v37);
}

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>
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> 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) {
  return 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>(v1, v2,
      v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18,
      v19, v20, v21, v22, v23, v24, v25, v26, v27, v28, v29, v30, v31, v32,
      v33, v34, v35, v36, v37, v38);
}

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>
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> 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) {
  return 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>(v1,
      v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17,
      v18, v19, v20, v21, v22, v23, v24, v25, v26, v27, v28, v29, v30, v31,
      v32, v33, v34, v35, v36, v37, v38, v39);
}

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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15,
      v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26, v27, v28, v29,
      v30, v31, v32, v33, v34, v35, v36, v37, v38, v39, v40);
}

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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14,
      v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26, v27, v28,
      v29, v30, v31, v32, v33, v34, v35, v36, v37, v38, v39, v40, v41);
}

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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13,
      v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26, v27,
      v28, v29, v30, v31, v32, v33, v34, v35, v36, v37, v38, v39, v40, v41,
      v42);
}

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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12,
      v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26,
      v27, v28, v29, v30, v31, v32, v33, v34, v35, v36, v37, v38, v39, v40,
      v41, v42, 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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11,
      v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25,
      v26, v27, v28, v29, v30, v31, v32, v33, v34, v35, v36, v37, v38, v39,
      v40, v41, v42, v43, 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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8, v9, v10,
      v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24,
      v25, v26, v27, v28, v29, v30, v31, v32, v33, v34, v35, v36, v37, v38,
      v39, v40, v41, v42, v43, v44, 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> 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8, v9,
      v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23,
      v24, v25, v26, v27, v28, v29, v30, v31, v32, v33, v34, v35, v36, v37,
      v38, v39, v40, v41, v42, v43, v44, v45, 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7, v8,
      v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23,
      v24, v25, v26, v27, v28, v29, v30, v31, v32, v33, v34, v35, v36, v37,
      v38, v39, v40, v41, v42, v43, v44, v45, v46, 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) {
  return 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>(v1, v2, v3, v4, v5, v6, v7,
      v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22,
      v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33, v34, v35, v36,
      v37, v38, v39, v40, v41, v42, v43, v44, v45, v46, v47, 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) {
  return 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>(v1, v2, v3, v4, v5, v6,
      v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21,
      v22, v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33, v34, v35,
      v36, v37, v38, v39, v40, v41, v42, v43, v44, v45, v46, v47, v48, 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) {
  return 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>(v1, v2, v3, v4,
      v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19,
      v20, v21, v22, v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33,
      v34, v35, v36, v37, v38, v39, v40, v41, v42, v43, v44, v45, v46, v47,
      v48, v49, v50);
}

// Bool() allows generating tests with parameters in a set of (false, true).
//
// Synopsis:
// Bool()
//   - returns a generator producing sequences with elements {false, true}.
//
// It is useful when testing code that depends on Boolean flags. Combinations
// of multiple flags can be tested when several Bool()'s are combined using
// Combine() function.
//
// In the following example all tests in the test case FlagDependentTest
// will be instantiated twice with parameters false and true.
//
// class FlagDependentTest : public testing::TestWithParam<bool> {
//   virtual void SetUp() {
//     external_flag = GetParam();
//   }
// }
// INSTANTIATE_TEST_CASE_P(BoolSequence, FlagDependentTest, Bool());
//
inline internal::ParamGenerator<bool> Bool() {
  return Values(false, true);
}

# if GTEST_HAS_COMBINE
// Combine() allows the user to combine two or more sequences to produce
// values of a Cartesian product of those sequences' elements.
//
// Synopsis:
// Combine(gen1, gen2, ..., genN)
//   - returns a generator producing sequences with elements coming from
//     the Cartesian product of elements from the sequences generated by
//     gen1, gen2, ..., genN. The sequence elements will have a type of
//     tuple<T1, T2, ..., TN> where T1, T2, ..., TN are the types
//     of elements from sequences produces by gen1, gen2, ..., genN.
//
// Combine can have up to 10 arguments. This number is currently limited
// by the maximum number of elements in the tuple implementation used by Google
// Test.
//
// Example:
//
// This will instantiate tests in test case AnimalTest each one with
// the parameter values tuple("cat", BLACK), tuple("cat", WHITE),
// tuple("dog", BLACK), and tuple("dog", WHITE):
//
// enum Color { BLACK, GRAY, WHITE };
// class AnimalTest
//     : public testing::TestWithParam<tuple<const char*, Color> > {...};
//
// TEST_P(AnimalTest, AnimalLooksNice) {...}
//
// INSTANTIATE_TEST_CASE_P(AnimalVariations, AnimalTest,
//                         Combine(Values("cat", "dog"),
//                                 Values(BLACK, WHITE)));
//
// This will instantiate tests in FlagDependentTest with all variations of two
// Boolean flags:
//
// class FlagDependentTest
//     : public testing::TestWithParam<tuple<bool, bool> > {
//   virtual void SetUp() {
//     // Assigns external_flag_1 and external_flag_2 values from the tuple.
//     tie(external_flag_1, external_flag_2) = GetParam();
//   }
// };
//
// TEST_P(FlagDependentTest, TestFeature1) {
//   // Test your code using external_flag_1 and external_flag_2 here.
// }
// INSTANTIATE_TEST_CASE_P(TwoBoolSequence, FlagDependentTest,
//                         Combine(Bool(), Bool()));
//
template <typename Generator1, typename Generator2>
internal::CartesianProductHolder2<Generator1, Generator2> Combine(
    const Generator1& g1, const Generator2& g2) {
  return internal::CartesianProductHolder2<Generator1, Generator2>(
      g1, g2);
}

template <typename Generator1, typename Generator2, typename Generator3>
internal::CartesianProductHolder3<Generator1, Generator2, Generator3> Combine(
    const Generator1& g1, const Generator2& g2, const Generator3& g3) {
  return internal::CartesianProductHolder3<Generator1, Generator2, Generator3>(
      g1, g2, g3);
}

template <typename Generator1, typename Generator2, typename Generator3,
    typename Generator4>
internal::CartesianProductHolder4<Generator1, Generator2, Generator3,
    Generator4> Combine(
    const Generator1& g1, const Generator2& g2, const Generator3& g3,
        const Generator4& g4) {
  return internal::CartesianProductHolder4<Generator1, Generator2, Generator3,
      Generator4>(
      g1, g2, g3, g4);
}

template <typename Generator1, typename Generator2, typename Generator3,
    typename Generator4, typename Generator5>
internal::CartesianProductHolder5<Generator1, Generator2, Generator3,
    Generator4, Generator5> Combine(
    const Generator1& g1, const Generator2& g2, const Generator3& g3,
        const Generator4& g4, const Generator5& g5) {
  return internal::CartesianProductHolder5<Generator1, Generator2, Generator3,
      Generator4, Generator5>(
      g1, g2, g3, g4, g5);
}

template <typename Generator1, typename Generator2, typename Generator3,
    typename Generator4, typename Generator5, typename Generator6>
internal::CartesianProductHolder6<Generator1, Generator2, Generator3,
    Generator4, Generator5, Generator6> Combine(
    const Generator1& g1, const Generator2& g2, const Generator3& g3,
        const Generator4& g4, const Generator5& g5, const Generator6& g6) {
  return internal::CartesianProductHolder6<Generator1, Generator2, Generator3,
      Generator4, Generator5, Generator6>(
      g1, g2, g3, g4, g5, g6);
}

template <typename Generator1, typename Generator2, typename Generator3,
    typename Generator4, typename Generator5, typename Generator6,
    typename Generator7>
internal::CartesianProductHolder7<Generator1, Generator2, Generator3,
    Generator4, Generator5, Generator6, Generator7> Combine(
    const Generator1& g1, const Generator2& g2, const Generator3& g3,
        const Generator4& g4, const Generator5& g5, const Generator6& g6,
        const Generator7& g7) {
  return internal::CartesianProductHolder7<Generator1, Generator2, Generator3,
      Generator4, Generator5, Generator6, Generator7>(
      g1, g2, g3, g4, g5, g6, g7);
}

template <typename Generator1, typename Generator2, typename Generator3,
    typename Generator4, typename Generator5, typename Generator6,
    typename Generator7, typename Generator8>
internal::CartesianProductHolder8<Generator1, Generator2, Generator3,
    Generator4, Generator5, Generator6, Generator7, Generator8> Combine(
    const Generator1& g1, const Generator2& g2, const Generator3& g3,
        const Generator4& g4, const Generator5& g5, const Generator6& g6,
        const Generator7& g7, const Generator8& g8) {
  return internal::CartesianProductHolder8<Generator1, Generator2, Generator3,
      Generator4, Generator5, Generator6, Generator7, Generator8>(
      g1, g2, g3, g4, g5, g6, g7, g8);
}

template <typename Generator1, typename Generator2, typename Generator3,
    typename Generator4, typename Generator5, typename Generator6,
    typename Generator7, typename Generator8, typename Generator9>
internal::CartesianProductHolder9<Generator1, Generator2, Generator3,
    Generator4, Generator5, Generator6, Generator7, Generator8,
    Generator9> Combine(
    const Generator1& g1, const Generator2& g2, const Generator3& g3,
        const Generator4& g4, const Generator5& g5, const Generator6& g6,
        const Generator7& g7, const Generator8& g8, const Generator9& g9) {
  return internal::CartesianProductHolder9<Generator1, Generator2, Generator3,
      Generator4, Generator5, Generator6, Generator7, Generator8, Generator9>(
      g1, g2, g3, g4, g5, g6, g7, g8, g9);
}

template <typename Generator1, typename Generator2, typename Generator3,
    typename Generator4, typename Generator5, typename Generator6,
    typename Generator7, typename Generator8, typename Generator9,
    typename Generator10>
internal::CartesianProductHolder10<Generator1, Generator2, Generator3,
    Generator4, Generator5, Generator6, Generator7, Generator8, Generator9,
    Generator10> Combine(
    const Generator1& g1, const Generator2& g2, const Generator3& g3,
        const Generator4& g4, const Generator5& g5, const Generator6& g6,
        const Generator7& g7, const Generator8& g8, const Generator9& g9,
        const Generator10& g10) {
  return internal::CartesianProductHolder10<Generator1, Generator2, Generator3,
      Generator4, Generator5, Generator6, Generator7, Generator8, Generator9,
      Generator10>(
      g1, g2, g3, g4, g5, g6, g7, g8, g9, g10);
}
# endif  // GTEST_HAS_COMBINE

# define TEST_P(test_case_name, test_name) \
  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()

// The optional last argument to INSTANTIATE_TEST_CASE_P allows the user
// to specify a function or functor that generates custom test name suffixes
// based on the test parameters. The function should accept one argument of
// type testing::TestParamInfo<class ParamType>, and return std::string.
//
// testing::PrintToStringParamName is a builtin test suffix generator that
// returns the value of testing::PrintToString(GetParam()).
//
// Note: test names must be non-empty, unique, and may only contain ASCII
// alphanumeric characters or underscore. Because PrintToString adds quotes
// to std::string and C strings, it won't work for these types.

# define INSTANTIATE_TEST_CASE_P(prefix, test_case_name, generator, ...) \
  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__)

}  // namespace testing

#endif  // GTEST_INCLUDE_GTEST_GTEST_PARAM_TEST_H_
```

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

```
#include <ostream>
#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.

```
// Copyright 2007, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


// Google Test - The Google C++ Testing and Mocking Framework
//
// This file implements a universal value printer that can print a
// value of any type T:
//
//   void ::testing::internal::UniversalPrinter<T>::Print(value, ostream_ptr);
//
// A user can teach this function how to print a class type T by
// defining either operator«() or PrintTo() in the namespace that
// defines T.  More specifically, the FIRST defined function in the
// following list will be used (assuming T is defined in namespace
// foo):
//
//   1. foo::PrintTo(const T&, ostream*)
//   2. operator«(ostream&, const T&) defined in either foo or the
//      global namespace.
//
// However if T is an STL-style container then it is printed element-wise
// unless foo::PrintTo(const T&, ostream*) is defined. Note that
// operator«() is ignored for container types.
//
// If none of the above is defined, it will print the debug string of
// the value if it is a protocol buffer, or print the raw bytes in the
// value otherwise.
//
// To aid debugging: when T is a reference type, the address of the
// value is also printed; when T is a (const) char pointer, both the
// pointer value and the NUL-terminated string it points to are
// printed.
//
// We also provide some convenient wrappers:
//
//   // Prints a value to a string.  For a (const or not) char
//   // pointer, the NUL-terminated string (but not the pointer) is
//   // printed.
//   std::string ::testing::PrintToString(const T& value);
//
//   // Prints a value tersely: for a reference type, the referenced
//   // value (but not the address) is printed; for a (const or not) char
//   // pointer, the NUL-terminated string (but not the pointer) is
//   // printed.
//   void ::testing::internal::UniversalTersePrint(const T& value, ostream*);
//
//   // Prints value using the type inferred by the compiler.  The difference
//   // from UniversalTersePrint() is that this function prints both the
//   // pointer and the NUL-terminated string for a (const or not) char pointer.
//   void ::testing::internal::UniversalPrint(const T& value, ostream*);
//
//   // Prints the fields of a tuple tersely to a string vector, one
//   // element for each field. Tuple support must be enabled in
//   // gtest-port.h.
//   std::vector<string> UniversalTersePrintTupleFieldsToStrings(
//       const Tuple& value);
//
// Known limitation:
//
// The print primitives print the elements of an STL-style container
// using the compiler-inferred type of *iter where iter is a
// const_iterator of the container.  When const_iterator is an input
// iterator but not a forward iterator, this inferred type may not
// match value_type, and the print output may be incorrect.  In
// practice, this is rarely a problem as for most containers
// const_iterator is a forward iterator.  We'll fix this if there's an
// actual need for it.  Note that this fix cannot rely on value_type
// being defined as many user-defined container types don't have
// value_type.

// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_GTEST_PRINTERS_H_
#define GTEST_INCLUDE_GTEST_GTEST_PRINTERS_H_

#include <ostream>  // NOLINT
#include <sstream>
#include <string>
#include <utility>
#include <vector>
#include "gtest/internal/gtest-port.h"
#include "gtest/internal/gtest-internal.h"

#if GTEST_HAS_STD_TUPLE_
# include <tuple>
#endif

#if GTEST_HAS_ABSL
#include "absl/strings/string_view.h"
#include "absl/types/optional.h"
#include "absl/types/variant.h"
#endif  // GTEST_HAS_ABSL

namespace testing {

// Definitions in the 'internal' and 'internal2' name spaces are
// subject to change without notice.  DO NOT USE THEM IN USER CODE!
namespace internal2 {

// Prints the given number of bytes in the given object to the given
// ostream.
GTEST_API_ void PrintBytesInObjectTo(const unsigned char* obj_bytes,
                                     size_t count,
                                     ::std::ostream* os);

// For selecting which printer to use when a given type has neither «
// nor PrintTo().
enum TypeKind {
  kProtobuf,              // a protobuf type
  kConvertibleToInteger,  // a type implicitly convertible to BiggestInt
                          // (e.g. a named or unnamed enum type)
#if GTEST_HAS_ABSL
  kConvertibleToStringView,  // a type implicitly convertible to
                             // absl::string_view
#endif
  kOtherType  // anything else
};

// TypeWithoutFormatter<T, kTypeKind>::PrintValue(value, os) is called
// by the universal printer to print a value of type T when neither
// operator« nor PrintTo() is defined for T, where kTypeKind is the
// "kind" of T as defined by enum TypeKind.
template <typename T, TypeKind kTypeKind>
class TypeWithoutFormatter {
 public:
  // This default version is called when kTypeKind is kOtherType.
  static void PrintValue(const T& value, ::std::ostream* os) {
    PrintBytesInObjectTo(static_cast<const unsigned char*>(
                             reinterpret_cast<const void*>(&value)),
                         sizeof(value), os);
  }
};

// We print a protobuf using its ShortDebugString() when the string
// doesn't exceed this many characters; otherwise we print it using
// DebugString() for better readability.
const size_t kProtobufOneLinerMaxLength = 50;

template <typename T>
class TypeWithoutFormatter<T, kProtobuf> {
 public:
  static void PrintValue(const T& value, ::std::ostream* os) {
    std::string pretty_str = value.ShortDebugString();
    if (pretty_str.length() > kProtobufOneLinerMaxLength) {
      pretty_str = "\n" + value.DebugString();
    }
    *os « ("<" + pretty_str + ">");
  }
};

template <typename T>
class TypeWithoutFormatter<T, kConvertibleToInteger> {
 public:
  // Since T has no « operator or PrintTo() but can be implicitly
  // converted to BiggestInt, we print it as a BiggestInt.
  //
  // Most likely T is an enum type (either named or unnamed), in which
  // case printing it as an integer is the desired behavior.  In case
  // T is not an enum, printing it as an integer is the best we can do
  // given that it has no user-defined printer.
  static void PrintValue(const T& value, ::std::ostream* os) {
    const internal::BiggestInt kBigInt = value;
    *os « kBigInt;
  }
};

#if GTEST_HAS_ABSL
template <typename T>
class TypeWithoutFormatter<T, kConvertibleToStringView> {
 public:
  // Since T has neither operator« nor PrintTo() but can be implicitly
  // converted to absl::string_view, we print it as a absl::string_view.
  //
  // Note: the implementation is further below, as it depends on
  // internal::PrintTo symbol which is defined later in the file.
  static void PrintValue(const T& value, ::std::ostream* os);
};
#endif

// Prints the given value to the given ostream.  If the value is a
// protocol message, its debug string is printed; if it's an enum or
// of a type implicitly convertible to BiggestInt, it's printed as an
// integer; otherwise the bytes in the value are printed.  This is
// what UniversalPrinter<T>::Print() does when it knows nothing about
// type T and T has neither « operator nor PrintTo().
//
// A user can override this behavior for a class type Foo by defining
// a « operator in the namespace where Foo is defined.
//
// We put this operator in namespace 'internal2' instead of 'internal'
// to simplify the implementation, as much code in 'internal' needs to
// use « in STL, which would conflict with our own « were it defined
// in 'internal'.
//
// Note that this operator« takes a generic std::basic_ostream<Char,
// CharTraits> type instead of the more restricted std::ostream.  If
// we define it to take an std::ostream instead, we'll get an
// "ambiguous overloads" compiler error when trying to print a type
// Foo that supports streaming to std::basic_ostream<Char,
// CharTraits>, as the compiler cannot tell whether
// operator«(std::ostream&, const T&) or
// operator«(std::basic_stream<Char, CharTraits>, const Foo&) is more
// specific.
template <typename Char, typename CharTraits, typename T>
::std::basic_ostream<Char, CharTraits>& operator«(
    ::std::basic_ostream<Char, CharTraits>& os, const T& x) {
  TypeWithoutFormatter<T, (internal::IsAProtocolMessage<T>::value
                               ? kProtobuf
                               : internal::ImplicitlyConvertible<
                                     const T&, internal::BiggestInt>::value
                                     ? kConvertibleToInteger
                                     :
#if GTEST_HAS_ABSL
                                     internal::ImplicitlyConvertible<
                                         const T&, absl::string_view>::value
                                         ? kConvertibleToStringView
                                         :
#endif
                                         kOtherType)>::PrintValue(x, &os);
  return os;
}

}  // namespace internal2
}  // namespace testing

// This namespace MUST NOT BE NESTED IN ::testing, or the name look-up
// magic needed for implementing UniversalPrinter won't work.
namespace testing_internal {

```

```
// Used to print a value that is not an STL-style container when the
// user doesn’t define PrintTo() for it.
template <typename T>
void DefaultPrintNonContainerTo(const T& value, ::std::ostream* os) {
  // With the following statement, during unqualified name lookup,
  // testing::internal2::operator« appears as if it was declared in
  // the nearest enclosing namespace that contains both
  // ::testing_internal and ::testing::internal2, i.e. the global
  // namespace.  For more details, refer to the C++ Standard section
  // 7.3.4-1 [namespace.udir].  This allows us to fall back onto
  // testing::internal2::operator« in case T doesn’t come with a «
  // operator.
  //
  // We cannot write ’using ::testing::internal2::operator«;’, which
  // gcc 3.3 fails to compile due to a compiler bug.
  using namespace ::testing::internal2;  // NOLINT

  // Assuming T is defined in namespace foo, in the next statement,
  // the compiler will consider all of:
  //
  //   1. foo::operator« (thanks to Koenig look-up),
  //   2. ::operator« (as the current namespace is enclosed in ::),
  //   3. testing::internal2::operator« (thanks to the using statement above).
  //
  // The operator« whose type matches T best will be picked.
  //
  // We deliberately allow #2 to be a candidate, as sometimes it’s
  // impossible to define #1 (e.g. when foo is ::std, defining
  // anything in it is undefined behavior unless you are a compiler
  // vendor.).
  *os « value;
}

}  // namespace testing_internal

namespace testing {
namespace internal {

// FormatForComparison<ToPrint, OtherOperand>::Format(value) formats a
// value of type ToPrint that is an operand of a comparison assertion
// (e.g. ASSERT_EQ).  OtherOperand is the type of the other operand in
// the comparison, and is used to help determine the best way to
// format the value.  In particular, when the value is a C string
// (char pointer) and the other operand is an STL string object, we
// want to format the C string as a string, since we know it is
// compared by value with the string object.  If the value is a char
// pointer but the other operand is not an STL string object, we don’t
// know whether the pointer is supposed to point to a NUL-terminated
// string, and thus want to print it as a pointer to be safe.
//
// INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.

// The default case.
template <typename ToPrint, typename OtherOperand>
class FormatForComparison {
 public:
  static ::std::string Format(const ToPrint& value) {
    return ::testing::PrintToString(value);
  }
};

// Array.
template <typename ToPrint, size_t N, typename OtherOperand>
class FormatForComparison<ToPrint[N], OtherOperand> {
 public:
  static ::std::string Format(const ToPrint* value) {
    return FormatForComparison<const ToPrint*, OtherOperand>::Format(value);
  }
};

// By default, print C string as pointers to be safe, as we don’t know
// whether they actually point to a NUL-terminated string.

#define GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_(CharType)                \
  template <typename OtherOperand>                                      \
  class FormatForComparison<CharType*, OtherOperand> {                  \
   public:                                                              \
    static ::std::string Format(CharType* value) {                      \
      return ::testing::PrintToString(static_cast<const void*>(value)); \
    }                                                                   \
  }

GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_(char);
GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_(const char);
GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_(wchar_t);
GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_(const wchar_t);

```

```
#undef GTEST_IMPL_FORMAT_C_STRING_AS_POINTER_

// If a C string is compared with an STL string object, we know it's meant
// to point to a NUL-terminated string, and thus can print it as a string.

#define GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(CharType, OtherStringType) \
  template <>                                                            \
  class FormatForComparison<CharType*, OtherStringType> {                \
   public:                                                               \
    static ::std::string Format(CharType* value) {                       \
      return ::testing::PrintToString(value);                            \
    }                                                                    \
  }

GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(char, ::std::string);
GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(const char, ::std::string);

#if GTEST_HAS_GLOBAL_STRING
GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(char, ::string);
GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(const char, ::string);
#endif

#if GTEST_HAS_GLOBAL_WSTRING
GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(wchar_t, ::wstring);
GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(const wchar_t, ::wstring);
#endif

#if GTEST_HAS_STD_WSTRING
GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(wchar_t, ::std::wstring);
GTEST_IMPL_FORMAT_C_STRING_AS_STRING_(const wchar_t, ::std::wstring);
#endif

#undef GTEST_IMPL_FORMAT_C_STRING_AS_STRING_

// Formats a comparison assertion (e.g. ASSERT_EQ, EXPECT_LT, and etc)
// operand to be used in a failure message.  The type (but not value)
// of the other operand may affect the format.  This allows us to
// print a char* as a raw pointer when it is compared against another
// char* or void*, and print it as a C string when it is compared
// against an std::string object, for example.
//
// INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
template <typename T1, typename T2>
std::string FormatForComparisonFailureMessage(
    const T1& value, const T2& /* other_operand */) {
  return FormatForComparison<T1, T2>::Format(value);
}

// UniversalPrinter<T>::Print(value, ostream_ptr) prints the given
// value to the given ostream.  The caller must ensure that
// 'ostream_ptr' is not NULL, or the behavior is undefined.
//
// We define UniversalPrinter as a class template (as opposed to a
// function template), as we need to partially specialize it for
// reference types, which cannot be done with function templates.
template <typename T>
class UniversalPrinter;

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

enum DefaultPrinterType {
  kPrintContainer,
  kPrintPointer,
  kPrintFunctionPointer,
  kPrintOther,
};
template <DefaultPrinterType type> struct WrapPrinterType {};

// Used to print an STL-style container when the user doesn't define
// a PrintTo() for it.
template <typename C>
void DefaultPrintTo(WrapPrinterType<kPrintContainer> /* dummy */,
                    const C& container, ::std::ostream* os) {
  const size_t kMaxCount = 32;  // The maximum number of elements to print.
  *os « '{';
  size_t count = 0;
  for (typename C::const_iterator it = container.begin();
       it != container.end(); ++it, ++count) {
    if (count > 0) {
      *os « ',';
      if (count == kMaxCount) {  // Enough has been printed.
        *os « " ...";
        break;
      }
    }
    *os « ' ';
    // We cannot call PrintTo(*it, os) here as PrintTo() doesn't
    // handle *it being a native array.
    internal::UniversalPrint(*it, os);
  }

  if (count > 0) {
    *os « ' ';
  }
  *os « '}';
}

// Used to print a pointer that is neither a char pointer nor a member
// pointer, when the user doesn't define PrintTo() for it.  (A member
// variable pointer or member function pointer doesn't really point to
// a location in the address space.  Their representation is
// implementation-defined.  Therefore they will be printed as raw
// bytes.)
template <typename T>
void DefaultPrintTo(WrapPrinterType<kPrintPointer> /* dummy */,
                    T* p, ::std::ostream* os) {
  if (p == NULL) {
    *os « "NULL";
  } else {
    // T is not a function type.  We just call « to print p,
    // relying on ADL to pick up user-defined « for their pointer
    // types, if any.
    *os « p;
  }
}
template <typename T>
void DefaultPrintTo(WrapPrinterType<kPrintFunctionPointer> /* dummy */,
                    T* p, ::std::ostream* os) {
  if (p == NULL) {
    *os « "NULL";
  } else {
    // T is a function type, so '*os « p' doesn't do what we want
    // (it just prints p as bool).  We want to print p as a const
    // void*.
    *os « reinterpret_cast<const void*>(p);
  }
}

// Used to print a non-container, non-pointer value when the user
// doesn't define PrintTo() for it.
template <typename T>
void DefaultPrintTo(WrapPrinterType<kPrintOther> /* dummy */,
                    const T& value, ::std::ostream* os) {
  ::testing_internal::DefaultPrintNonContainerTo(value, os);
}

// Prints the given value using the « operator if it has one;
// otherwise prints the bytes in it.  This is what
// UniversalPrinter<T>::Print() does when PrintTo() is not specialized
// or overloaded for type T.
//
// A user can override this behavior for a class type Foo by defining
// an overload of PrintTo() in the namespace where Foo is defined.  We
// give the user this option as sometimes defining a « operator for
// Foo is not desirable (e.g. the coding style may prevent doing it,
// or there is already a « operator but it doesn't do what the user
// wants).
template <typename T>
void PrintTo(const T& value, ::std::ostream* os) {
  // DefaultPrintTo() is overloaded.  The type of its first argument
  // determines which version will be picked.
  //
  // Note that we check for container types here, prior to we check
  // for protocol message types in our operator«.  The rationale is:
  //
  // For protocol messages, we want to give people a chance to
  // override Google Mock's format by defining a PrintTo() or
  // operator«.  For STL containers, other formats can be
  // incompatible with Google Mock's format for the container
  // elements; therefore we check for container types here to ensure
  // that our format is used.
  //
  // Note that MSVC and clang-cl do allow an implicit conversion from
  // pointer-to-function to pointer-to-object, but clang-cl warns on it.
  // So don't use ImplicitlyConvertible if it can be helped since it will
  // cause this warning, and use a separate overload of DefaultPrintTo for
  // function pointers so that the `*os « p' in the object pointer overload
  // doesn't cause that warning either.
  DefaultPrintTo(
      WrapPrinterType <
                  (sizeof(IsContainerTest<T>(0)) == sizeof(IsContainer)) &&
              !IsRecursiveContainer<T>::value
          ? kPrintContainer
          : !is_pointer<T>::value
                  ? kPrintOther
#if GTEST_LANG_CXX11
                  : std::is_function<typename std::remove_pointer<T>::type>::value
#else
                  : !internal::ImplicitlyConvertible<T, const void*>::value
#endif
                        ? kPrintFunctionPointer
                        : kPrintPointer > (),
      value, os);
}

// The following list of PrintTo() overloads tells
// UniversalPrinter<T>::Print() how to print standard types (built-in
// types, strings, plain arrays, and pointers).

// Overloads for various char types.
GTEST_API_ void PrintTo(unsigned char c, ::std::ostream* os);
GTEST_API_ void PrintTo(signed char c, ::std::ostream* os);
inline void PrintTo(char c, ::std::ostream* os) {
  // When printing a plain char, we always treat it as unsigned.  This
  // way, the output won't be affected by whether the compiler thinks
  // char is signed or not.
  PrintTo(static_cast<unsigned char>(c), os);
}

// Overloads for other simple built-in types.
inline void PrintTo(bool x, ::std::ostream* os) {
  *os « (x ? "true" : "false");
}

// Overload for wchar_t type.
// Prints a wchar_t as a symbol if it is printable or as its internal
// code otherwise and also as its decimal code (except for L'\0').
// The L'\0' char is printed as "L'\\0'". The decimal code is printed
// as signed integer when wchar_t is implemented by the compiler
// as a signed type and is printed as an unsigned integer when wchar_t
// is implemented as an unsigned type.
GTEST_API_ void PrintTo(wchar_t wc, ::std::ostream* os);

// Overloads for C strings.
GTEST_API_ void PrintTo(const char* s, ::std::ostream* os);
inline void PrintTo(char* s, ::std::ostream* os) {
  PrintTo(ImplicitCast_<const char*>(s), os);
}

// signed/unsigned char is often used for representing binary data, so
// we print pointers to it as void* to be safe.
inline void PrintTo(const signed char* s, ::std::ostream* os) {
  PrintTo(ImplicitCast_<const void*>(s), os);
}
inline void PrintTo(signed char* s, ::std::ostream* os) {
  PrintTo(ImplicitCast_<const void*>(s), os);
}
inline void PrintTo(const unsigned char* s, ::std::ostream* os) {
  PrintTo(ImplicitCast_<const void*>(s), os);
}
inline void PrintTo(unsigned char* s, ::std::ostream* os) {
  PrintTo(ImplicitCast_<const void*>(s), os);
}

// MSVC can be configured to define wchar_t as a typedef of unsigned
// short.  It defines _NATIVE_WCHAR_T_DEFINED when wchar_t is a native
// type.  When wchar_t is a typedef, defining an overload for const
// wchar_t* would cause unsigned short* be printed as a wide string,
// possibly causing invalid memory accesses.
#if !defined(_MSC_VER) || defined(_NATIVE_WCHAR_T_DEFINED)
// Overloads for wide C strings
GTEST_API_ void PrintTo(const wchar_t* s, ::std::ostream* os);
inline void PrintTo(wchar_t* s, ::std::ostream* os) {
  PrintTo(ImplicitCast_<const wchar_t*>(s), os);
}
#endif

// Overload for C arrays.  Multi-dimensional arrays are printed
// properly.

// Prints the given number of elements in an array, without printing
// the curly braces.
template <typename T>
void PrintRawArrayTo(const T a[], size_t count, ::std::ostream* os) {
  UniversalPrint(a[0], os);
  for (size_t i = 1; i != count; i++) {
    *os « ", ";
    UniversalPrint(a[i], os);
  }
}
```

```

// Overloads for ::string and ::std::string.
#if GTEST_HAS_GLOBAL_STRING
GTEST_API_ void PrintStringTo(const ::string&s, ::std::ostream* os);
inline void PrintTo(const ::string& s, ::std::ostream* os) {
  PrintStringTo(s, os);
}
#endif  // GTEST_HAS_GLOBAL_STRING

GTEST_API_ void PrintStringTo(const ::std::string&s, ::std::ostream* os);
inline void PrintTo(const ::std::string& s, ::std::ostream* os) {
  PrintStringTo(s, os);
}

// Overloads for ::wstring and ::std::wstring.
#if GTEST_HAS_GLOBAL_WSTRING
GTEST_API_ void PrintWideStringTo(const ::wstring&s, ::std::ostream* os);
inline void PrintTo(const ::wstring& s, ::std::ostream* os) {
  PrintWideStringTo(s, os);
}
#endif  // GTEST_HAS_GLOBAL_WSTRING

#if GTEST_HAS_STD_WSTRING
GTEST_API_ void PrintWideStringTo(const ::std::wstring&s, ::std::ostream* os);
inline void PrintTo(const ::std::wstring& s, ::std::ostream* os) {
  PrintWideStringTo(s, os);
}
#endif  // GTEST_HAS_STD_WSTRING

#if GTEST_HAS_ABSL
// Overload for absl::string_view.
inline void PrintTo(absl::string_view sp, ::std::ostream* os) {
  PrintTo(::std::string(sp), os);
}
#endif  // GTEST_HAS_ABSL

#if GTEST_LANG_CXX11
inline void PrintTo(std::nullptr_t, ::std::ostream* os) { *os « "(nullptr)"; }
#endif  // GTEST_LANG_CXX11

#if GTEST_HAS_TR1_TUPLE || GTEST_HAS_STD_TUPLE_
// Helper function for printing a tuple.  T must be instantiated with
// a tuple type.
template <typename T>
void PrintTupleTo(const T& t, ::std::ostream* os);
#endif  // GTEST_HAS_TR1_TUPLE || GTEST_HAS_STD_TUPLE_

#if GTEST_HAS_TR1_TUPLE
// Overload for ::std::tr1::tuple.  Needed for printing function arguments,
// which are packed as tuples.

// Overloaded PrintTo() for tuples of various arities.  We support
// tuples of up-to 10 fields.  The following implementation works
// regardless of whether tr1::tuple is implemented using the
// non-standard variadic template feature or not.

inline void PrintTo(const ::std::tr1::tuple<>& t, ::std::ostream* os) {
  PrintTupleTo(t, os);
}

template <typename T1>
void PrintTo(const ::std::tr1::tuple<T1>& t, ::std::ostream* os) {
  PrintTupleTo(t, os);
}

template <typename T1, typename T2>
void PrintTo(const ::std::tr1::tuple<T1, T2>& t, ::std::ostream* os) {
  PrintTupleTo(t, os);
}

template <typename T1, typename T2, typename T3>
void PrintTo(const ::std::tr1::tuple<T1, T2, T3>& t, ::std::ostream* os) {
  PrintTupleTo(t, os);
}

template <typename T1, typename T2, typename T3, typename T4>
void PrintTo(const ::std::tr1::tuple<T1, T2, T3, T4>& t, ::std::ostream* os) {
  PrintTupleTo(t, os);
}

template <typename T1, typename T2, typename T3, typename T4, typename T5>
void PrintTo(const ::std::tr1::tuple<T1, T2, T3, T4, T5>& t,
             ::std::ostream* os) {
  PrintTupleTo(t, os);
}

template <typename T1, typename T2, typename T3, typename T4, typename T5,
          typename T6>
void PrintTo(const ::std::tr1::tuple<T1, T2, T3, T4, T5, T6>& t,
             ::std::ostream* os) {
  PrintTupleTo(t, os);
}

template <typename T1, typename T2, typename T3, typename T4, typename T5,
          typename T6, typename T7>
void PrintTo(const ::std::tr1::tuple<T1, T2, T3, T4, T5, T6, T7>& t,
             ::std::ostream* os) {
  PrintTupleTo(t, os);
}

template <typename T1, typename T2, typename T3, typename T4, typename T5,
          typename T6, typename T7, typename T8>
void PrintTo(const ::std::tr1::tuple<T1, T2, T3, T4, T5, T6, T7, T8>& t,
             ::std::ostream* os) {
  PrintTupleTo(t, os);
}

template <typename T1, typename T2, typename T3, typename T4, typename T5,
          typename T6, typename T7, typename T8, typename T9>
void PrintTo(const ::std::tr1::tuple<T1, T2, T3, T4, T5, T6, T7, T8, T9>& t,
             ::std::ostream* os) {
  PrintTupleTo(t, os);
}

template <typename T1, typename T2, typename T3, typename T4, typename T5,
          typename T6, typename T7, typename T8, typename T9, typename T10>
void PrintTo(
    const ::std::tr1::tuple<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10>& t,
    ::std::ostream* os) {
  PrintTupleTo(t, os);
}
#endif  // GTEST_HAS_TR1_TUPLE

#if GTEST_HAS_STD_TUPLE_
template <typename... Types>
void PrintTo(const ::std::tuple<Types...>& t, ::std::ostream* os) {
  PrintTupleTo(t, os);
}
#endif  // GTEST_HAS_STD_TUPLE_

// Overload for std::pair.
template <typename T1, typename T2>
void PrintTo(const ::std::pair<T1, T2>& value, ::std::ostream* os) {
  *os « '(';
  // We cannot use UniversalPrint(value.first, os) here, as T1 may be
  // a reference type.  The same for printing value.second.
  UniversalPrinter<T1>::Print(value.first, os);
  *os « ", ";
  UniversalPrinter<T2>::Print(value.second, os);
  *os « ')';
}

// Implements printing a non-reference type T by letting the compiler
// pick the right overload of PrintTo() for T.
template <typename T>
class UniversalPrinter {
 public:
  // MSVC warns about adding const to a function type, so we want to
  // disable the warning.
  GTEST_DISABLE_MSC_WARNINGS_PUSH_(4180)

  // Note: we deliberately don't call this PrintTo(), as that name
  // conflicts with ::testing::internal::PrintTo in the body of the
  // function.
  static void Print(const T& value, ::std::ostream* os) {
    // By default, ::testing::internal::PrintTo() is used for printing
    // the value.
    //
    // Thanks to Koenig look-up, if T is a class and has its own
    // PrintTo() function defined in its namespace, that function will
    // be visible here.  Since it is more specific than the generic ones
    // in ::testing::internal, it will be picked by the compiler in the
    // following statement - exactly what we want.
    PrintTo(value, os);
  }

  GTEST_DISABLE_MSC_WARNINGS_POP_()
};

#if GTEST_HAS_ABSL

// Printer for absl::optional

template <typename T>
```

```
class UniversalPrinter<::absl::optional<T» {
 public:
  static void Print(const ::absl::optional<T>& value, ::std::ostream* os) {
    *os « '(';
    if (!value) {
      *os « "nullopt";
    } else {
      UniversalPrint(*value, os);
    }
    *os « ')';
  }
};

// Printer for absl::variant

template <typename... T>
class UniversalPrinter<::absl::variant<T...» {
 public:
  static void Print(const ::absl::variant<T...>& value, ::std::ostream* os) {
    *os « '(';
    absl::visit(Visitor{os}, value);
    *os « ')';
  }

 private:
  struct Visitor {
    template <typename U>
    void operator()(const U& u) const {
      *os « "'" « GetTypeName<U>() « "' with value ";
      UniversalPrint(u, os);
    }
    ::std::ostream* os;
  };
};

#endif  // GTEST_HAS_ABSL

// UniversalPrintArray(begin, len, os) prints an array of 'len'
// elements, starting at address 'begin'.
template <typename T>
void UniversalPrintArray(const T* begin, size_t len, ::std::ostream* os) {
  if (len == 0) {
    *os « "{}";
  } else {
    *os « "{ ";
    const size_t kThreshold = 18;
    const size_t kChunkSize = 8;
    // If the array has more than kThreshold elements, we'll have to
    // omit some details by printing only the first and the last
    // kChunkSize elements.
    // FIXME: let the user control the threshold using a flag.
    if (len <= kThreshold) {
      PrintRawArrayTo(begin, len, os);
    } else {
      PrintRawArrayTo(begin, kChunkSize, os);
      *os « ", ..., ";
      PrintRawArrayTo(begin + len - kChunkSize, kChunkSize, os);
    }
    *os « " }";
  }
}
// This overload prints a (const) char array compactly.
GTEST_API_ void UniversalPrintArray(
    const char* begin, size_t len, ::std::ostream* os);

// This overload prints a (const) wchar_t array compactly.
GTEST_API_ void UniversalPrintArray(
    const wchar_t* begin, size_t len, ::std::ostream* os);

// Implements printing an array type T[N].
template <typename T, size_t N>
class UniversalPrinter<T[N]> {
 public:
  // Prints the given array, omitting some elements when there are too
  // many.
  static void Print(const T (&a)[N], ::std::ostream* os) {
    UniversalPrintArray(a, N, os);
  }
};

// Implements printing a reference type T&.
template <typename T>
class UniversalPrinter<T&> {
 public:
  // MSVC warns about adding const to a function type, so we want to
  // disable the warning.
  GTEST_DISABLE_MSC_WARNINGS_PUSH_(4180)

  static void Print(const T& value, ::std::ostream* os) {
    // Prints the address of the value.  We use reinterpret_cast here
    // as static_cast doesn't compile when T is a function type.
    *os « "@" « reinterpret_cast<const void*>(&value) « " ";

    // Then prints the value itself.
    UniversalPrint(value, os);
  }

  GTEST_DISABLE_MSC_WARNINGS_POP_()
};

// Prints a value tersely: for a reference type, the referenced value
// (but not the address) is printed; for a (const) char pointer, the
// NUL-terminated string (but not the pointer) is printed.

template <typename T>
class UniversalTersePrinter {
 public:
  static void Print(const T& value, ::std::ostream* os) {
    UniversalPrint(value, os);
  }
};
template <typename T>
class UniversalTersePrinter<T&> {
 public:
  static void Print(const T& value, ::std::ostream* os) {
    UniversalPrint(value, os);
  }
};
template <typename T, size_t N>
class UniversalTersePrinter<T[N]> {
 public:
  static void Print(const T (&value)[N], ::std::ostream* os) {
    UniversalPrinter<T[N]>::Print(value, os);
  }
};
template <>
class UniversalTersePrinter<const char*> {
 public:
  static void Print(const char* str, ::std::ostream* os) {
    if (str == NULL) {
      *os « "NULL";
    } else {
      UniversalPrint(std::string(str), os);
    }
  }
};
template <>
class UniversalTersePrinter<char*> {
 public:
  static void Print(char* str, ::std::ostream* os) {
    UniversalTersePrinter<const char*>::Print(str, os);
  }
};

#if GTEST_HAS_STD_WSTRING
template <>
class UniversalTersePrinter<const wchar_t*> {
 public:
  static void Print(const wchar_t* str, ::std::ostream* os) {
    if (str == NULL) {
      *os « "NULL";
    } else {
      UniversalPrint(::std::wstring(str), os);
    }
  }
};
#endif

template <>
class UniversalTersePrinter<wchar_t*> {
 public:
  static void Print(wchar_t* str, ::std::ostream* os) {
    UniversalTersePrinter<const wchar_t*>::Print(str, os);
  }
};

template <typename T>
void UniversalTersePrint(const T& value, ::std::ostream* os) {
  UniversalTersePrinter<T>::Print(value, os);
}

// Prints a value using the type inferred by the compiler.  The
// difference between this and UniversalTersePrint() is that for a
// (const) char pointer, this prints both the pointer and the
// NUL-terminated string.
template <typename T>
void UniversalPrint(const T& value, ::std::ostream* os) {
  // A workarond for the bug in VC++ 7.1 that prevents us from instantiating
  // UniversalPrinter with T directly.
  typedef T T1;
  UniversalPrinter<T1>::Print(value, os);
}

typedef ::std::vector< ::std::string> Strings;

// TuplePolicy<TupleT> must provide:
// - tuple_size
//     size of tuple TupleT.
// - get<size_t I>(const TupleT& t)
//     static function extracting element I of tuple TupleT.
// - tuple_element<size_t I>::type
//     type of element I of tuple TupleT.
template <typename TupleT>
struct TuplePolicy;

#if GTEST_HAS_TR1_TUPLE
template <typename TupleT>
struct TuplePolicy {
  typedef TupleT Tuple;
  static const size_t tuple_size = ::std::tr1::tuple_size<Tuple>::value;

  template <size_t I>
  struct tuple_element : ::std::tr1::tuple_element<static_cast<int>(I), Tuple> {
  };

  template <size_t I>
  static typename AddReference<const typename ::std::tr1::tuple_element<
      static_cast<int>(I), Tuple>::type>::type
  get(const Tuple& tuple) {
    return ::std::tr1::get<I>(tuple);
  }
};
template <typename TupleT>
const size_t TuplePolicy<TupleT>::tuple_size;
#endif  // GTEST_HAS_TR1_TUPLE

#if GTEST_HAS_STD_TUPLE_
template <typename... Types>
struct TuplePolicy< ::std::tuple<Types...> > {
  typedef ::std::tuple<Types...> Tuple;
  static const size_t tuple_size = ::std::tuple_size<Tuple>::value;

  template <size_t I>
  struct tuple_element : ::std::tuple_element<I, Tuple> {};

  template <size_t I>
  static const typename ::std::tuple_element<I, Tuple>::type& get(
      const Tuple& tuple) {
    return ::std::get<I>(tuple);
  }
};
template <typename... Types>
const size_t TuplePolicy< ::std::tuple<Types...> >::tuple_size;
#endif  // GTEST_HAS_STD_TUPLE_

#if GTEST_HAS_TR1_TUPLE || GTEST_HAS_STD_TUPLE_
// This helper template allows PrintTo() for tuples and
// UniversalTersePrintTupleFieldsToStrings() to be defined by
// induction on the number of tuple fields.  The idea is that
// TuplePrefixPrinter<N>::PrintPrefixTo(t, os) prints the first N
// fields in tuple t, and can be defined in terms of
// TuplePrefixPrinter<N - 1>.
//
// The inductive case.
template <size_t N>
struct TuplePrefixPrinter {
  // Prints the first N fields of a tuple.
  template <typename Tuple>
  static void PrintPrefixTo(const Tuple& t, ::std::ostream* os) {
    TuplePrefixPrinter<N - 1>::PrintPrefixTo(t, os);
    GTEST_INTENTIONAL_CONST_COND_PUSH_()
    if (N > 1) {
    GTEST_INTENTIONAL_CONST_COND_POP_()
      *os « ", ";
    }
    UniversalPrinter<
        typename TuplePolicy<Tuple>::template tuple_element<N - 1>::type>
        ::Print(TuplePolicy<Tuple>::template get<N - 1>(t), os);
  }

  // Tersely prints the first N fields of a tuple to a string vector,
  // one element for each field.
  template <typename Tuple>
  static void TersePrintPrefixToStrings(const Tuple& t, Strings* strings) {
    TuplePrefixPrinter<N - 1>::TersePrintPrefixToStrings(t, strings);
    ::std::stringstream ss;
    UniversalTersePrint(TuplePolicy<Tuple>::template get<N - 1>(t), &ss);
    strings->push_back(ss.str());
  }
};

// Base case.
template <>
struct TuplePrefixPrinter<0> {
  template <typename Tuple>
  static void PrintPrefixTo(const Tuple&, ::std::ostream*) {}

  template <typename Tuple>
  static void TersePrintPrefixToStrings(const Tuple&, Strings*) {}
};

// Helper function for printing a tuple.
// Tuple must be either std::tr1::tuple or std::tuple type.
template <typename Tuple>
void PrintTupleTo(const Tuple& t, ::std::ostream* os) {
  *os « "(";
  TuplePrefixPrinter<TuplePolicy<Tuple>::tuple_size>::PrintPrefixTo(t, os);
  *os « ")";
}

// Prints the fields of a tuple tersely to a string vector, one
// element for each field.  See the comment before
// UniversalTersePrint() for how we define "tersely".
template <typename Tuple>
Strings UniversalTersePrintTupleFieldsToStrings(const Tuple& value) {
  Strings result;
  TuplePrefixPrinter<TuplePolicy<Tuple>::tuple_size>::
      TersePrintPrefixToStrings(value, &result);
  return result;
}
#endif  // GTEST_HAS_TR1_TUPLE || GTEST_HAS_STD_TUPLE_

}  // namespace internal

#if GTEST_HAS_ABSL
namespace internal2 {
template <typename T>
void TypeWithoutFormatter<T, kConvertibleToStringView>::PrintValue(
    const T& value, ::std::ostream* os) {
  internal::PrintTo(absl::string_view(value), os);
}
}  // namespace internal2
#endif

template <typename T>
::std::string PrintToString(const T& value) {
  ::std::stringstream ss;
  internal::UniversalTersePrinter<T>::Print(value, &ss);
  return ss.str();
}

}  // namespace testing

// Include any custom printer added by the local installation.
// We must include this header at the end to make sure it can use the
// declarations from this file.
#include "gtest/internal/custom/gtest-printers.h"

#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.

```
// Copyright 2015, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// This file provides an injection point for custom printers in a local
// installation of gTest.
// It will be included from gtest-printers.h and the overrides in this file
// will be visible to everyone.
//
// Injection point for custom user configurations. See README for details
//
// ** Custom implementation starts here **

#ifndef GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_PRINTERS_H_
#define GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_PRINTERS_H_

#endif  // GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_PRINTERS_H_
```

9.14 Dokumentacja pliku packages/Microsoft.googletest.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.

```
// Copyright 2007, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

//
// Utilities for testing Google Test itself and code that uses Google Test
// (e.g. frameworks built on top of Google Test).

// GOOGLETEST_CM0004 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_GTEST_SPI_H_
#define GTEST_INCLUDE_GTEST_GTEST_SPI_H_

#include "gtest/gtest.h"
```

```

GTEST_DISABLE_MSC_WARNINGS_PUSH_(4251 \
/* class A needs to have dll-interface to be used by clients of class B */)

namespace testing {

// This helper class can be used to mock out Google Test failure reporting
// so that we can test Google Test or code that builds on Google Test.
//
// An object of this class appends a TestPartResult object to the
// TestPartResultArray object given in the constructor whenever a Google Test
// failure is reported. It can either intercept only failures that are
// generated in the same thread that created this object or it can intercept
// all generated failures. The scope of this mock object can be controlled with
// the second argument to the two arguments constructor.
class GTEST_API_ ScopedFakeTestPartResultReporter
    : public TestPartResultReporterInterface {
 public:
  // The two possible mocking modes of this object.
  enum InterceptMode {
    INTERCEPT_ONLY_CURRENT_THREAD,  // Intercepts only thread local failures.
    INTERCEPT_ALL_THREADS           // Intercepts all failures.
  };

  // The c'tor sets this object as the test part result reporter used
  // by Google Test.  The 'result' parameter specifies where to report the
  // results. This reporter will only catch failures generated in the current
  // thread. DEPRECATED
  explicit ScopedFakeTestPartResultReporter(TestPartResultArray* result);

  // Same as above, but you can choose the interception scope of this object.
  ScopedFakeTestPartResultReporter(InterceptMode intercept_mode,
                                   TestPartResultArray* result);

  // The d'tor restores the previous test part result reporter.
  virtual ~ScopedFakeTestPartResultReporter();

  // Appends the TestPartResult object to the TestPartResultArray
  // received in the constructor.
  //
  // This method is from the TestPartResultReporterInterface
  // interface.
  virtual void ReportTestPartResult(const TestPartResult& result);
 private:
  void Init();

  const InterceptMode intercept_mode_;
  TestPartResultReporterInterface* old_reporter_;
  TestPartResultArray* const result_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(ScopedFakeTestPartResultReporter);
};

namespace internal {

// A helper class for implementing EXPECT_FATAL_FAILURE() and
// EXPECT_NONFATAL_FAILURE().  Its destructor verifies that the given
// TestPartResultArray contains exactly one failure that has the given
// type and contains the given substring.  If that's not the case, a
// non-fatal failure will be generated.
class GTEST_API_ SingleFailureChecker {
 public:
  // The constructor remembers the arguments.
  SingleFailureChecker(const TestPartResultArray* results,
                       TestPartResult::Type type, const std::string& substr);
  ~SingleFailureChecker();
 private:
  const TestPartResultArray* const results_;
  const TestPartResult::Type type_;
  const std::string substr_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(SingleFailureChecker);
};

}  // namespace internal

}  // namespace testing

GTEST_DISABLE_MSC_WARNINGS_POP_()  //  4251

// A set of macros for testing Google Test assertions or code that's expected
// to generate Google Test fatal failures.  It verifies that the given
// statement will cause exactly one fatal Google Test failure with 'substr'
// being part of the failure message.
//
// There are two different versions of this macro. EXPECT_FATAL_FAILURE only
// affects and considers failures generated in the current thread and
// EXPECT_FATAL_FAILURE_ON_ALL_THREADS does the same but for all threads.
//
// The verification of the assertion is done correctly even when the statement
// throws an exception or aborts the current function.
//
// Known restrictions:
//   - 'statement' cannot reference local non-static variables or
//     non-static members of the current object.
//   - 'statement' cannot return a value.
//   - You cannot stream a failure message to this macro.
//
// Note that even though the implementations of the following two
// macros are much alike, we cannot refactor them to use a common
// helper macro, due to some peculiarity in how the preprocessor
// works.  The AcceptsMacroThatExpandsToUnprotectedComma test in
// gtest_unittest.cc will fail to compile if we do that.
#define EXPECT_FATAL_FAILURE(statement, substr) \
  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())

#define EXPECT_FATAL_FAILURE_ON_ALL_THREADS(statement, substr) \
  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())

// A macro for testing Google Test assertions or code that's expected to
// generate Google Test non-fatal failures.  It asserts that the given
// statement will cause exactly one non-fatal Google Test failure with 'substr'
// being part of the failure message.
//
// There are two different versions of this macro. EXPECT_NONFATAL_FAILURE only
// affects and considers failures generated in the current thread and
// EXPECT_NONFATAL_FAILURE_ON_ALL_THREADS does the same but for all threads.
//
// 'statement' is allowed to reference local variables and members of
// the current object.
//
// The verification of the assertion is done correctly even when the statement
// throws an exception or aborts the current function.
//
// Known restrictions:
//   - You cannot stream a failure message to this macro.
//
// Note that even though the implementations of the following two
// macros are much alike, we cannot refactor them to use a common
// helper macro, due to some peculiarity in how the preprocessor
// works.  If we do that, the code won't compile when the user gives
// EXPECT_NONFATAL_FAILURE() a statement that contains a macro that
// expands to code containing an unprotected comma.  The
// AcceptsMacroThatExpandsToUnprotectedComma test in gtest_unittest.cc
// catches that.
//
// For the same reason, we have to write
//   if (::testing::internal::AlwaysTrue()) { statement; }
// instead of
//   GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement)
// to avoid an MSVC warning on unreachable code.
#define EXPECT_NONFATAL_FAILURE(statement, substr) \
  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())

#define EXPECT_NONFATAL_FAILURE_ON_ALL_THREADS(statement, substr) \
  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())

#endif  // GTEST_INCLUDE_GTEST_GTEST_SPI_H_
```

9.16 Dokumentacja pliku packages/Microsoft.googletest.v140.windesktop.msvcstl.static.rt-dyn.1.8.1.8/build/native/include/gtest/gtest-test-part.h

```
#include <iosfwd>
#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.

```
// Copyright 2008, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_GTEST_TEST_PART_H_
#define GTEST_INCLUDE_GTEST_GTEST_TEST_PART_H_

#include <iosfwd>
#include <vector>
#include "gtest/internal/gtest-internal.h"
#include "gtest/internal/gtest-string.h"

GTEST_DISABLE_MSC_WARNINGS_PUSH_(4251 \
/* class A needs to have dll-interface to be used by clients of class B */)

namespace testing {

// A copyable object representing the result of a test part (i.e. an
// assertion or an explicit FAIL(), ADD_FAILURE(), or SUCCESS()).
//
// Don't inherit from TestPartResult as its destructor is not virtual.
class GTEST_API_ TestPartResult {
 public:
  // The possible outcomes of a test part (i.e. an assertion or an
  // explicit SUCCEED(), FAIL(), or ADD_FAILURE()).
  enum Type {
    kSuccess,          // Succeeded.
    kNonFatalFailure,  // Failed but the test can continue.
    kFatalFailure      // Failed and the test should be terminated.
  };

  // C'tor.  TestPartResult does NOT have a default constructor.
  // Always use this constructor (with parameters) to create a
  // TestPartResult object.
  TestPartResult(Type a_type,
                 const char* a_file_name,
                 int a_line_number,
                 const char* a_message)
      : type_(a_type),
        file_name_(a_file_name == NULL ? "" : a_file_name),
        line_number_(a_line_number),
        summary_(ExtractSummary(a_message)),
        message_(a_message) {
  }

  // Gets the outcome of the test part.
  Type type() const { return type_; }

  // Gets the name of the source file where the test part took place, or
  // NULL if it's unknown.
  const char* file_name() const {
    return file_name_.empty() ? NULL : file_name_.c_str();
  }

  // Gets the line in the source file where the test part took place,
  // or -1 if it's unknown.
  int line_number() const { return line_number_; }

  // Gets the summary of the failure message.
  const char* summary() const { return summary_.c_str(); }

  // Gets the message associated with the test part.
  const char* message() const { return message_.c_str(); }

  // Returns true iff the test part passed.
  bool passed() const { return type_ == kSuccess; }

  // Returns true iff the test part failed.
  bool failed() const { return type_ != kSuccess; }

  // Returns true iff the test part non-fatally failed.
  bool nonfatally_failed() const { return type_ == kNonFatalFailure; }

  // Returns true iff the test part fatally failed.
  bool fatally_failed() const { return type_ == kFatalFailure; }

 private:
  Type type_;

  // Gets the summary of the failure message by omitting the stack
  // trace in it.
  static std::string ExtractSummary(const char* message);

  // The name of the source file where the test part took place, or
  // "" if the source file is unknown.
  std::string file_name_;
  // The line in the source file where the test part took place, or -1
  // if the line number is unknown.
  int line_number_;
  std::string summary_;  // The test failure summary.
  std::string message_;  // The test failure message.
};

// Prints a TestPartResult object.
std::ostream& operator«(std::ostream& os, const TestPartResult& result);

// An array of TestPartResult objects.
//
// Don't inherit from TestPartResultArray as its destructor is not
// virtual.
class GTEST_API_ TestPartResultArray {
 public:
  TestPartResultArray() {}

  // Appends the given TestPartResult to the array.
  void Append(const TestPartResult& result);

  // Returns the TestPartResult at the given index (0-based).
  const TestPartResult& GetTestPartResult(int index) const;

  // Returns the number of TestPartResult objects in the array.
  int size() const;

 private:
  std::vector<TestPartResult> array_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(TestPartResultArray);
};

// This interface knows how to report a test part result.
class GTEST_API_ TestPartResultReporterInterface {
 public:
  virtual ~TestPartResultReporterInterface() {}

  virtual void ReportTestPartResult(const TestPartResult& result) = 0;
};

namespace internal {

// This helper class is used by {ASSERT|EXPECT}_NO_FATAL_FAILURE to check if a
// statement generates new fatal failures. To do so it registers itself as the
// current test part result reporter. Besides checking if fatal failures were
// reported, it only delegates the reporting to the former result reporter.
// The original result reporter is restored in the destructor.
// INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
class GTEST_API_ HasNewFatalFailureHelper
    : public TestPartResultReporterInterface {
 public:
  HasNewFatalFailureHelper();
  virtual ~HasNewFatalFailureHelper();
  virtual void ReportTestPartResult(const TestPartResult& result);
  bool has_new_fatal_failure() const { return has_new_fatal_failure_; }
 private:
  bool has_new_fatal_failure_;
  TestPartResultReporterInterface* original_reporter_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(HasNewFatalFailureHelper);
};

}  // namespace internal

}  // namespace testing

GTEST_DISABLE_MSC_WARNINGS_POP_()  //  4251

#endif  // GTEST_INCLUDE_GTEST_GTEST_TEST_PART_H_
```

9.18 Dokumentacja pliku packages/Microsoft.googletest.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.

```
// Copyright 2008 Google Inc.
// All Rights Reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_GTEST_TYPED_TEST_H_
#define GTEST_INCLUDE_GTEST_GTEST_TYPED_TEST_H_

// This header implements typed tests and type-parameterized tests.

// Typed (aka type-driven) tests repeat the same test for types in a
// list.  You must know which types you want to test with when writing
// typed tests. Here's how you do it:

#if 0

// First, define a fixture class template.  It should be parameterized
// by a type.  Remember to derive it from testing::Test.
template <typename T>
class FooTest : public testing::Test {
 public:
  ...
  typedef std::list<T> List;
  static T shared_;
  T value_;
};

// Next, associate a list of types with the test case, which will be
// repeated for each type in the list.  The typedef is necessary for
// the macro to parse correctly.
typedef testing::Types<char, int, unsigned int> MyTypes;
TYPED_TEST_CASE(FooTest, MyTypes);

// If the type list contains only one type, you can write that type
// directly without Types<...>:
//   TYPED_TEST_CASE(FooTest, int);

// Then, use TYPED_TEST() instead of TEST_F() to define as many typed
// tests for this test case as you want.
TYPED_TEST(FooTest, DoesBlah) {
  // Inside a test, refer to TypeParam to get the type parameter.
  // Since we are inside a derived class template, C++ requires use to
  // visit the members of FooTest via 'this'.
  TypeParam n = this->value_;

  // To visit static members of the fixture, add the TestFixture::
  // prefix.
  n += TestFixture::shared_;

  // To refer to typedefs in the fixture, add the "typename
  // TestFixture::" prefix.
  typename TestFixture::List values;
  values.push_back(n);
  ...
}

TYPED_TEST(FooTest, HasPropertyA) { ... }

// TYPED_TEST_CASE takes an optional third argument which allows to specify a
// class that generates custom test name suffixes based on the type. This should
// be a class which has a static template function GetName(int index) returning
// a string for each type. The provided integer index equals the index of the
// type in the provided type list. In many cases the index can be ignored.
//
// For example:
//   class MyTypeNames {
//    public:
//     template <typename T>
//     static std::string GetName(int) {
//       if (std::is_same<T, char>()) return "char";
//       if (std::is_same<T, int>()) return "int";
//       if (std::is_same<T, unsigned int>()) return "unsignedInt";
//     }
//   };
//   TYPED_TEST_CASE(FooTest, MyTypes, MyTypeNames);

#endif  // 0

// Type-parameterized tests are abstract test patterns parameterized
// by a type.  Compared with typed tests, type-parameterized tests
// allow you to define the test pattern without knowing what the type
// parameters are.  The defined pattern can be instantiated with
// different types any number of times, in any number of translation
// units.
//
// If you are designing an interface or concept, you can define a
// suite of type-parameterized tests to verify properties that any
// valid implementation of the interface/concept should have.  Then,
// each implementation can easily instantiate the test suite to verify
// that it conforms to the requirements, without having to write
// similar tests repeatedly.  Here's an example:

#if 0

// First, define a fixture class template.  It should be parameterized
// by a type.  Remember to derive it from testing::Test.
template <typename T>
class FooTest : public testing::Test {
  ...
};

// Next, declare that you will define a type-parameterized test case
// (the _P suffix is for "parameterized" or "pattern", whichever you
// prefer):
TYPED_TEST_CASE_P(FooTest);

// Then, use TYPED_TEST_P() to define as many type-parameterized tests
// for this type-parameterized test case as you want.
TYPED_TEST_P(FooTest, DoesBlah) {
  // Inside a test, refer to TypeParam to get the type parameter.
  TypeParam n = 0;
  ...
}

TYPED_TEST_P(FooTest, HasPropertyA) { ... }

// Now the tricky part: you need to register all test patterns before
// you can instantiate them.  The first argument of the macro is the
// test case name; the rest are the names of the tests in this test
// case.
REGISTER_TYPED_TEST_CASE_P(FooTest,
                           DoesBlah, HasPropertyA);

// Finally, you are free to instantiate the pattern with the types you
// want.  If you put the above code in a header file, you can #include
// it in multiple C++ source files and instantiate it multiple times.
//
// To distinguish different instances of the pattern, the first
// argument to the INSTANTIATE_* macro is a prefix that will be added
// to the actual test case name.  Remember to pick unique prefixes for
// different instances.
typedef testing::Types<char, int, unsigned int> MyTypes;
INSTANTIATE_TYPED_TEST_CASE_P(My, FooTest, MyTypes);

// If the type list contains only one type, you can write that type
// directly without Types<...>:
//   INSTANTIATE_TYPED_TEST_CASE_P(My, FooTest, int);
//
// Similar to the optional argument of TYPED_TEST_CASE above,
// INSTANTIATE_TEST_CASE_P takes an optional fourth argument which allows to
// generate custom names.
//   INSTANTIATE_TYPED_TEST_CASE_P(My, FooTest, MyTypes, MyTypeNames);

#endif  // 0

#include "gtest/internal/gtest-port.h"
#include "gtest/internal/gtest-type-util.h"

// Implements typed tests.

#if GTEST_HAS_TYPED_TEST

// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
//
// Expands to the name of the typedef for the type parameters of the
// given test case.
# define GTEST_TYPE_PARAMS_(TestCaseName) gtest_type_params_##TestCaseName##_

// Expands to the name of the typedef for the NameGenerator, responsible for
// creating the suffixes of the name.
#define GTEST_NAME_GENERATOR_(TestCaseName) \
  gtest_type_params_##TestCaseName##_NameGenerator

// The 'Types' template argument below must have spaces around it
// since some compilers may choke on '»' when passing a template
// instance (e.g. Types<int>)
# define TYPED_TEST_CASE(CaseName, Types, ...)                             \
  typedef ::testing::internal::TypeList< Types >::type GTEST_TYPE_PARAMS_( \
      CaseName);                                                           \
  typedef ::testing::internal::NameGeneratorSelector<__VA_ARGS__>::type    \
      GTEST_NAME_GENERATOR_(CaseName)

# define TYPED_TEST(CaseName, TestName)                                       \
  template <typename gtest_TypeParam_>                                        \
  class GTEST_TEST_CLASS_NAME_(CaseName, TestName)                            \
      : public CaseName<gtest_TypeParam_> {                                   \
   private:                                                                   \
    typedef CaseName<gtest_TypeParam_> TestFixture;                           \
    typedef gtest_TypeParam_ TypeParam;                                       \
    virtual void TestBody();                                                  \
  };                                                                          \
  static bool gtest_##CaseName##_##TestName##_registered_                     \
        GTEST_ATTRIBUTE_UNUSED_ =                                             \
      ::testing::internal::TypeParameterizedTest<                             \
          CaseName,                                                           \
          ::testing::internal::TemplateSel<GTEST_TEST_CLASS_NAME_(CaseName,   \
                                                                  TestName)>, \
          GTEST_TYPE_PARAMS_(                                                 \
              CaseName)>::Register("",                                        \
                                   ::testing::internal::CodeLocation(         \
                                       __FILE__, __LINE__),                   \
                                   #CaseName, #TestName, 0,                   \
                                   ::testing::internal::GenerateNames<        \
                                       GTEST_NAME_GENERATOR_(CaseName),       \
                                       GTEST_TYPE_PARAMS_(CaseName)>());      \
  template <typename gtest_TypeParam_>                                        \
  void GTEST_TEST_CLASS_NAME_(CaseName,                                       \
                              TestName)<gtest_TypeParam_>::TestBody()

#endif  // GTEST_HAS_TYPED_TEST

// Implements type-parameterized tests.

#if GTEST_HAS_TYPED_TEST_P

// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
//
// Expands to the namespace name that the type-parameterized tests for
// the given type-parameterized test case are defined in.  The exact
// name of the namespace is subject to change without notice.
# define GTEST_CASE_NAMESPACE_(TestCaseName) \
  gtest_case_##TestCaseName##_

// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
//
// Expands to the name of the variable used to remember the names of
// the defined tests in the given test case.
# define GTEST_TYPED_TEST_CASE_P_STATE_(TestCaseName) \
  gtest_typed_test_case_p_state_##TestCaseName##_

// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE DIRECTLY.
//
// Expands to the name of the variable used to remember the names of
// the registered tests in the given test case.
# define GTEST_REGISTERED_TEST_NAMES_(TestCaseName) \
  gtest_registered_test_names_##TestCaseName##_

// The variables defined in the type-parameterized test macros are
// static as typically these macros are used in a .h file that can be
// #included in multiple translation units linked together.
# define TYPED_TEST_CASE_P(CaseName) \
  static ::testing::internal::TypedTestCasePState \
      GTEST_TYPED_TEST_CASE_P_STATE_(CaseName)

# define TYPED_TEST_P(CaseName, TestName) \
  namespace GTEST_CASE_NAMESPACE_(CaseName) { \
  template <typename gtest_TypeParam_> \
  class TestName : public CaseName<gtest_TypeParam_> { \
   private: \
    typedef CaseName<gtest_TypeParam_> TestFixture; \
    typedef gtest_TypeParam_ TypeParam; \
    virtual void TestBody(); \
  }; \
  static bool gtest_##TestName##_defined_ GTEST_ATTRIBUTE_UNUSED_ = \
      GTEST_TYPED_TEST_CASE_P_STATE_(CaseName).AddTestName(\
          __FILE__, __LINE__, #CaseName, #TestName); \
  } \
  template <typename gtest_TypeParam_> \
  void GTEST_CASE_NAMESPACE_(CaseName)::TestName<gtest_TypeParam_>::TestBody()

# define REGISTER_TYPED_TEST_CASE_P(CaseName, ...) \
  namespace GTEST_CASE_NAMESPACE_(CaseName) { \
  typedef ::testing::internal::Templates<__VA_ARGS__>::type gtest_AllTests_; \
  } \
  static const char* const GTEST_REGISTERED_TEST_NAMES_(CaseName) \
      GTEST_ATTRIBUTE_UNUSED_ = \
          GTEST_TYPED_TEST_CASE_P_STATE_(CaseName).VerifyRegisteredTestNames( \
              __FILE__, __LINE__, #__VA_ARGS__)

// The 'Types' template argument below must have spaces around it
// since some compilers may choke on '>>' when passing a template
// instance (e.g. Types<int>)
# define INSTANTIATE_TYPED_TEST_CASE_P(Prefix, CaseName, Types, ...)      \
  static bool gtest_##Prefix##_##CaseName GTEST_ATTRIBUTE_UNUSED_ =       \
      ::testing::internal::TypeParameterizedTestCase<                     \
          CaseName, GTEST_CASE_NAMESPACE_(CaseName)::gtest_AllTests_,     \
          ::testing::internal::TypeList< Types >::type>::                 \
          Register(#Prefix,                                               \
                   ::testing::internal::CodeLocation(__FILE__, __LINE__), \
                   &GTEST_TYPED_TEST_CASE_P_STATE_(CaseName), #CaseName,  \
                   GTEST_REGISTERED_TEST_NAMES_(CaseName),                \
                   ::testing::internal::GenerateNames<                    \
                       ::testing::internal::NameGeneratorSelector<        \
                           __VA_ARGS__>::type,                            \
                       ::testing::internal::TypeList< Types >::type>())

#endif  // GTEST_HAS_TYPED_TEST_P

#endif  // GTEST_INCLUDE_GTEST_GTEST_TYPED_TEST_H_
```

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

```
#include <limits>
#include <ostream>
#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::IsSubstring (const char ∗needle_expr, const char ∗haystack_expr, const char ∗needle, const char ∗haystack)
- GTEST_API_ AssertionResult testing::IsSubstring (const char ∗needle_expr, const char ∗haystack_expr, const wchar_t ∗needle, const wchar_t ∗haystack)
- GTEST_API_ AssertionResult testing::IsNotSubstring (const char ∗needle_expr, const char ∗haystack_expr, const char ∗needle, const char ∗haystack)
- GTEST_API_ AssertionResult testing::IsNotSubstring (const char ∗needle_expr, const char ∗haystack_expr, const wchar_t ∗needle, const wchar_t ∗haystack)
- GTEST_API_ AssertionResult testing::IsSubstring (const char ∗needle_expr, const char ∗haystack_expr, const ::std::string &needle, const ::std::string &haystack)
- GTEST_API_ AssertionResult testing::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::FloatLE (const char ∗expr1, const char ∗expr2, float val1, float val2)
- GTEST_API_ AssertionResult testing::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 testing::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)
```

Wartość:

```
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::GetTestTypeId())
```

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.

```
// Copyright 2005, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

//
// The Google C++ Testing and Mocking Framework (Google Test)
//
// This header file defines the public API for Google Test.  It should be
// included by any test program that uses Google Test.
//
// IMPORTANT NOTE: Due to limitation of the C++ language, we have to
// leave some internal implementation details in this header file.
// They are clearly marked by comments like this:
//
//   // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
//
// Such code is NOT meant to be used by a user directly, and is subject
// to CHANGE WITHOUT NOTICE.  Therefore DO NOT DEPEND ON IT in a user
// program!
//
// Acknowledgment: Google Test borrowed the idea of automatic test
// registration from Barthelemy Dagenais' (barthelemy@prologique.com)
// easyUnit framework.

// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_GTEST_H_
#define GTEST_INCLUDE_GTEST_GTEST_H_

#include <limits>
#include <ostream>
#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"

GTEST_DISABLE_MSC_WARNINGS_PUSH_(4251 \
/* class A needs to have dll-interface to be used by clients of class B */)

// Depending on the platform, different string classes are available.
// On Linux, in addition to ::std::string, Google also makes use of
// class ::string, which has the same interface as ::std::string, but
// has a different implementation.
//
// You can define GTEST_HAS_GLOBAL_STRING to 1 to indicate that
// ::string is available AND is a distinct type to ::std::string, or
// define it to 0 to indicate otherwise.
//
// If ::std::string and ::string are the same class on your platform
// due to aliasing, you should define GTEST_HAS_GLOBAL_STRING to 0.
//
// If you do not define GTEST_HAS_GLOBAL_STRING, it is defined
// heuristically.

namespace testing {

// Silence C4100 (unreferenced formal parameter) and 4805
// unsafe mix of type 'const int' and type 'const bool'
#ifdef _MSC_VER
# pragma warning(push)
# pragma warning(disable:4805)
# pragma warning(disable:4100)
#endif


// Declares the flags.

// This flag temporary enables the disabled tests.
GTEST_DECLARE_bool_(also_run_disabled_tests);

// This flag brings the debugger on an assertion failure.
GTEST_DECLARE_bool_(break_on_failure);

// This flag controls whether Google Test catches all test-thrown exceptions
// and logs them as failures.
GTEST_DECLARE_bool_(catch_exceptions);

// This flag enables using colors in terminal output. Available values are
// "yes" to enable colors, "no" (disable colors), or "auto" (the default)
// to let Google Test decide.
GTEST_DECLARE_string_(color);

// This flag sets up the filter to select by name using a glob pattern
// the tests to run. If the filter is not given all tests are executed.
GTEST_DECLARE_string_(filter);

// This flag controls whether Google Test installs a signal handler that dumps
// debugging information when fatal signals are raised.
GTEST_DECLARE_bool_(install_failure_signal_handler);

// This flag causes the Google Test to list tests. None of the tests listed
// are actually run if the flag is provided.
GTEST_DECLARE_bool_(list_tests);

// This flag controls whether Google Test emits a detailed XML report to a file
// in addition to its normal textual output.
GTEST_DECLARE_string_(output);

// This flags control whether Google Test prints the elapsed time for each
// test.
GTEST_DECLARE_bool_(print_time);

// This flags control whether Google Test prints UTF8 characters as text.
GTEST_DECLARE_bool_(print_utf8);

// This flag specifies the random number seed.
GTEST_DECLARE_int32_(random_seed);

// This flag sets how many times the tests are repeated. The default value
// is 1. If the value is -1 the tests are repeating forever.
GTEST_DECLARE_int32_(repeat);

// This flag controls whether Google Test includes Google Test internal
// stack frames in failure stack traces.
GTEST_DECLARE_bool_(show_internal_stack_frames);

// When this flag is specified, tests' order is randomized on every iteration.
GTEST_DECLARE_bool_(shuffle);

// This flag specifies the maximum number of stack frames to be
// printed in a failure message.
GTEST_DECLARE_int32_(stack_trace_depth);

// When this flag is specified, a failed assertion will throw an
// exception if exceptions are enabled, or exit the program with a
// non-zero code otherwise. For use with an external test framework.
GTEST_DECLARE_bool_(throw_on_failure);

// When this flag is set with a "host:port" string, on supported
// platforms test results are streamed to the specified port on
// the specified host machine.
GTEST_DECLARE_string_(stream_result_to);

#if GTEST_USE_OWN_FLAGFILE_FLAG_
GTEST_DECLARE_string_(flagfile);
#endif  // GTEST_USE_OWN_FLAGFILE_FLAG_

// The upper limit for valid stack trace depths.
const int kMaxStackTraceDepth = 100;

namespace internal {

class AssertHelper;
class DefaultGlobalTestPartResultReporter;
class ExecDeathTest;
class NoExecDeathTest;
class FinalSuccessChecker;
class GTestFlagSaver;
class StreamingListenerTest;
class TestResultAccessor;
class TestEventListenersAccessor;
class TestEventRepeater;
class UnitTestRecordPropertyTestHelper;
class WindowsDeathTest;
class FuchsiaDeathTest;
class UnitTestImpl* GetUnitTestImpl();
void ReportFailureInUnknownLocation(TestPartResult::Type result_type,
                                    const std::string& message);

}  // namespace internal

// The friend relationship of some of these classes is cyclic.
// If we don't forward declare them the compiler might confuse the classes
// in friendship clauses with same named classes on the scope.
class Test;
class TestCase;
class TestInfo;
class UnitTest;

// A class for indicating whether an assertion was successful.  When
// the assertion wasn't successful, the AssertionResult object
// remembers a non-empty message that describes how it failed.
//
// To create an instance of this class, use one of the factory functions
// (AssertionSuccess() and AssertionFailure()).
//
// This class is useful for two purposes:
//   1. Defining predicate functions to be used with Boolean test assertions
//      EXPECT_TRUE/EXPECT_FALSE and their ASSERT_ counterparts
//   2. Defining predicate-format functions to be
//      used with predicate assertions (ASSERT_PRED_FORMAT*, etc).
//
// For example, if you define IsEven predicate:
//
//   testing::AssertionResult IsEven(int n) {
//     if ((n % 2) == 0)
//       return testing::AssertionSuccess();
//     else
//       return testing::AssertionFailure() « n « " is odd";
//   }
//
// Then the failed expectation EXPECT_TRUE(IsEven(Fib(5)))
// will print the message
//
//   Value of: IsEven(Fib(5))
//     Actual: false (5 is odd)
//   Expected: true
//
// instead of a more opaque
//
```

```
//   Value of: IsEven(Fib(5))
//     Actual: false
//   Expected: true
//
// in case IsEven is a simple Boolean predicate.
//
// If you expect your predicate to be reused and want to support informative
// messages in EXPECT_FALSE and ASSERT_FALSE (negative assertions show up
// about half as often as positive ones in our tests), supply messages for
// both success and failure cases:
//
//   testing::AssertionResult IsEven(int n) {
//     if ((n % 2) == 0)
//       return testing::AssertionSuccess() « n « " is even";
//     else
//       return testing::AssertionFailure() « n « " is odd";
//   }
//
// Then a statement EXPECT_FALSE(IsEven(Fib(6))) will print
//
//   Value of: IsEven(Fib(6))
//     Actual: true (8 is even)
//   Expected: false
//
// NB: Predicates that support negative Boolean assertions have reduced
// performance in positive ones so be careful not to use them in tests
// that have lots (tens of thousands) of positive Boolean assertions.
//
// To use this class with EXPECT_PRED_FORMAT assertions such as:
//
//   // Verifies that Foo() returns an even number.
//   EXPECT_PRED_FORMAT1(IsEven, Foo());
//
// you need to define:
//
//   testing::AssertionResult IsEven(const char* expr, int n) {
//     if ((n % 2) == 0)
//       return testing::AssertionSuccess();
//     else
//       return testing::AssertionFailure()
//         « "Expected: " « expr « " is even\n  Actual: it's " « n;
//   }
//
// If Foo() returns 5, you will see the following message:
//
//   Expected: Foo() is even
//     Actual: it's 5
//
class GTEST_API_ AssertionResult {
 public:
  // Copy constructor.
  // Used in EXPECT_TRUE/FALSE(assertion_result).
  AssertionResult(const AssertionResult& other);

#if defined(_MSC_VER) && _MSC_VER < 1910
  GTEST_DISABLE_MSC_WARNINGS_PUSH_(4800 /* forcing value to bool */)
#endif

  // Used in the EXPECT_TRUE/FALSE(bool_expression).
  //
  // T must be contextually convertible to bool.
  //
  // The second parameter prevents this overload from being considered if
  // the argument is implicitly convertible to AssertionResult. In that case
  // we want AssertionResult's copy constructor to be used.
  template <typename T>
  explicit AssertionResult(
      const T& success,
      typename internal::EnableIf<
          !internal::ImplicitlyConvertible<T, AssertionResult>::value>::type*
          /*enabler*/ = NULL)
      : success_(success) {}

#if defined(_MSC_VER) && _MSC_VER < 1910
  GTEST_DISABLE_MSC_WARNINGS_POP_()
#endif

  // Assignment operator.
  AssertionResult& operator=(AssertionResult other) {
    swap(other);
    return *this;
  }

  // Returns true iff the assertion succeeded.
  operator bool() const { return success_; }  // NOLINT

  // Returns the assertion's negation. Used with EXPECT/ASSERT_FALSE.
  AssertionResult operator!() const;

  // Returns the text streamed into this AssertionResult. Test assertions
  // use it when they fail (i.e., the predicate's outcome doesn't match the
  // assertion's expectation). When nothing has been streamed into the
  // object, returns an empty string.
  const char* message() const {
    return message_.get() != NULL ?  message_->c_str() : "";
  }
  // FIXME: Remove this after making sure no clients use it.
  // Deprecated; please use message() instead.
  const char* failure_message() const { return message(); }

  // Streams a custom failure message into this object.
  template <typename T> AssertionResult& operator<<(const T& value) {
    AppendMessage(Message() << value);
    return *this;
  }

  // Allows streaming basic output manipulators such as endl or flush into
  // this object.
  AssertionResult& operator<<(
      ::std::ostream& (*basic_manipulator)(::std::ostream& stream)) {
    AppendMessage(Message() << basic_manipulator);
    return *this;
  }

 private:
  // Appends the contents of message to message_.
  void AppendMessage(const Message& a_message) {
    if (message_.get() == NULL)
      message_.reset(new ::std::string);
    message_->append(a_message.GetString().c_str());
  }

  // Swap the contents of this AssertionResult with other.
  void swap(AssertionResult& other);

  // Stores result of the assertion predicate.
  bool success_;
  // Stores the message describing the condition in case the expectation
  // construct is not satisfied with the predicate's outcome.
  // Referenced via a pointer to avoid taking too much stack frame space
  // with test assertions.
  internal::scoped_ptr< ::std::string> message_;
};

// Makes a successful assertion result.
GTEST_API_ AssertionResult AssertionSuccess();

// Makes a failed assertion result.
GTEST_API_ AssertionResult AssertionFailure();

// Makes a failed assertion result with the given failure message.
// Deprecated; use AssertionFailure() << msg.
GTEST_API_ AssertionResult AssertionFailure(const Message& msg);

}  // namespace testing

// Includes the auto-generated header that implements a family of generic
// predicate assertion macros. This include comes late because it relies on
// APIs declared above.
#include "gtest/gtest_pred_impl.h"

namespace testing {

// The abstract class that all tests inherit from.
//
// In Google Test, a unit test program contains one or many TestCases, and
// each TestCase contains one or many Tests.
//
// When you define a test using the TEST macro, you don't need to
// explicitly derive from Test - the TEST macro automatically does
// this for you.
//
// The only time you derive from Test is when defining a test fixture
// to be used in a TEST_F.  For example:
//
//   class FooTest : public testing::Test {
//    protected:
//     void SetUp() override { ... }
//     void TearDown() override { ... }
//     ...
//   };
//
//   TEST_F(FooTest, Bar) { ... }
//   TEST_F(FooTest, Baz) { ... }
```

```
//
// Test is not copyable.
class GTEST_API_ Test {
 public:
  friend class TestInfo;

  // Defines types for pointers to functions that set up and tear down
  // a test case.
  typedef internal::SetUpTestCaseFunc SetUpTestCaseFunc;
  typedef internal::TearDownTestCaseFunc TearDownTestCaseFunc;

  // The d'tor is virtual as we intend to inherit from Test.
  virtual ~Test();

  // Sets up the stuff shared by all tests in this test case.
  //
  // Google Test will call Foo::SetUpTestCase() before running the first
  // test in test case Foo.  Hence a sub-class can define its own
  // SetUpTestCase() method to shadow the one defined in the super
  // class.
  static void SetUpTestCase() {}

  // Tears down the stuff shared by all tests in this test case.
  //
  // Google Test will call Foo::TearDownTestCase() after running the last
  // test in test case Foo.  Hence a sub-class can define its own
  // TearDownTestCase() method to shadow the one defined in the super
  // class.
  static void TearDownTestCase() {}

  // Returns true iff the current test has a fatal failure.
  static bool HasFatalFailure();

  // Returns true iff the current test has a non-fatal failure.
  static bool HasNonfatalFailure();

  // Returns true iff the current test has a (either fatal or
  // non-fatal) failure.
  static bool HasFailure() { return HasFatalFailure() || HasNonfatalFailure(); }

  // Logs a property for the current test, test case, or for the entire
  // invocation of the test program when used outside of the context of a
  // test case.  Only the last value for a given key is remembered.  These
  // are public static so they can be called from utility functions that are
  // not members of the test fixture.  Calls to RecordProperty made during
  // lifespan of the test (from the moment its constructor starts to the
  // moment its destructor finishes) will be output in XML as attributes of
  // the <testcase> element.  Properties recorded from fixture's
  // SetUpTestCase or TearDownTestCase are logged as attributes of the
  // corresponding <testsuite> element.  Calls to RecordProperty made in the
  // global context (before or after invocation of RUN_ALL_TESTS and from
  // SetUp/TearDown method of Environment objects registered with Google
  // Test) will be output as attributes of the <testsuites> element.
  static void RecordProperty(const std::string& key, const std::string& value);
  static void RecordProperty(const std::string& key, int value);

 protected:
  // Creates a Test object.
  Test();

  // Sets up the test fixture.
  virtual void SetUp();

  // Tears down the test fixture.
  virtual void TearDown();

 private:
  // Returns true iff the current test has the same fixture class as
  // the first test in the current test case.
  static bool HasSameFixtureClass();

  // Runs the test after the test fixture has been set up.
  //
  // A sub-class must implement this to define the test logic.
  //
  // DO NOT OVERRIDE THIS FUNCTION DIRECTLY IN A USER PROGRAM.
  // Instead, use the TEST or TEST_F macro.
  virtual void TestBody() = 0;

  // Sets up, executes, and tears down the test.
  void Run();

  // Deletes self.  We deliberately pick an unusual name for this
  // internal method to avoid clashing with names used in user TESTs.
  void DeleteSelf_() { delete this; }

  const internal::scoped_ptr< GTEST_FLAG_SAVER_ > gtest_flag_saver_;

  // Often a user misspells SetUp() as Setup() and spends a long time
  // wondering why it is never called by Google Test.  The declaration of
  // the following method is solely for catching such an error at
  // compile time:
  //
  //   - The return type is deliberately chosen to be not void, so it
  //   will be a conflict if void Setup() is declared in the user's
  //   test fixture.
  //
  //   - This method is private, so it will be another compiler error
  //   if the method is called from the user's test fixture.
  //
  // DO NOT OVERRIDE THIS FUNCTION.
  //
  // If you see an error about overriding the following function or
  // about it being private, you have mis-spelled SetUp() as Setup().
  struct Setup_should_be_spelled_SetUp {};
  virtual Setup_should_be_spelled_SetUp* Setup() { return NULL; }

  // We disallow copying Tests.
  GTEST_DISALLOW_COPY_AND_ASSIGN_(Test);
};

typedef internal::TimeInMillis TimeInMillis;

// A copyable object representing a user specified test property which can be
// output as a key/value string pair.
//
// Don't inherit from TestProperty as its destructor is not virtual.
class TestProperty {
 public:
  // C'tor.  TestProperty does NOT have a default constructor.
  // Always use this constructor (with parameters) to create a
  // TestProperty object.
  TestProperty(const std::string& a_key, const std::string& a_value) :
    key_(a_key), value_(a_value) {
  }

  // Gets the user supplied key.
  const char* key() const {
    return key_.c_str();
  }

  // Gets the user supplied value.
  const char* value() const {
    return value_.c_str();
  }

  // Sets a new value, overriding the one supplied in the constructor.
  void SetValue(const std::string& new_value) {
    value_ = new_value;
  }

 private:
  // The key supplied by the user.
  std::string key_;
  // The value supplied by the user.
  std::string value_;
};

// The result of a single Test.  This includes a list of
// TestPartResults, a list of TestProperties, a count of how many
// death tests there are in the Test, and how much time it took to run
// the Test.
//
// TestResult is not copyable.
class GTEST_API_ TestResult {
 public:
  // Creates an empty TestResult.
  TestResult();

  // D'tor.  Do not inherit from TestResult.
  ~TestResult();

  // Gets the number of all test parts.  This is the sum of the number
  // of successful test parts and the number of failed test parts.
  int total_part_count() const;

  // Returns the number of the test properties.
  int test_property_count() const;

  // Returns true iff the test passed (i.e. no test part failed).
  bool Passed() const { return !Failed(); }

  // Returns true iff the test failed.
  bool Failed() const;

  // Returns true iff the test fatally failed.
  bool HasFatalFailure() const;

  // Returns true iff the test has a non-fatal failure.
  bool HasNonfatalFailure() const;

  // Returns the elapsed time, in milliseconds.
  TimeInMillis elapsed_time() const { return elapsed_time_; }

  // Returns the i-th test part result among all the results. i can range from 0
  // to total_part_count() - 1. If i is not in that range, aborts the program.
  const TestPartResult& GetTestPartResult(int i) const;

  // Returns the i-th test property. i can range from 0 to
  // test_property_count() - 1. If i is not in that range, aborts the
  // program.
  const TestProperty& GetTestProperty(int i) const;

 private:
  friend class TestInfo;
  friend class TestCase;
  friend class UnitTest;
  friend class internal::DefaultGlobalTestPartResultReporter;
  friend class internal::ExecDeathTest;
  friend class internal::TestResultAccessor;
  friend class internal::UnitTestImpl;
  friend class internal::WindowsDeathTest;
  friend class internal::FuchsiaDeathTest;

  // Gets the vector of TestPartResults.
  const std::vector<TestPartResult>& test_part_results() const {
    return test_part_results_;
  }

  // Gets the vector of TestProperties.
  const std::vector<TestProperty>& test_properties() const {
    return test_properties_;
  }

  // Sets the elapsed time.
  void set_elapsed_time(TimeInMillis elapsed) { elapsed_time_ = elapsed; }

  // Adds a test property to the list. The property is validated and may add
  // a non-fatal failure if invalid (e.g., if it conflicts with reserved
  // key names). If a property is already recorded for the same key, the
  // value will be updated, rather than storing multiple values for the same
  // key.  xml_element specifies the element for which the property is being
  // recorded and is used for validation.
  void RecordProperty(const std::string& xml_element,
                      const TestProperty& test_property);

  // Adds a failure if the key is a reserved attribute of Google Test
  // testcase tags.  Returns true if the property is valid.
  // FIXME: Validate attribute names are legal and human readable.
  static bool ValidateTestProperty(const std::string& xml_element,
                                   const TestProperty& test_property);

  // Adds a test part result to the list.
  void AddTestPartResult(const TestPartResult& test_part_result);

  // Returns the death test count.
  int death_test_count() const { return death_test_count_; }

  // Increments the death test count, returning the new count.
  int increment_death_test_count() { return ++death_test_count_; }

  // Clears the test part results.
  void ClearTestPartResults();

  // Clears the object.
  void Clear();

  // Protects mutable state of the property vector and of owned
  // properties, whose values may be updated.
  internal::Mutex test_properites_mutex_;

  // The vector of TestPartResults
  std::vector<TestPartResult> test_part_results_;
  // The vector of TestProperties
  std::vector<TestProperty> test_properties_;
  // Running count of death tests.
  int death_test_count_;
  // The elapsed time, in milliseconds.
  TimeInMillis elapsed_time_;

  // We disallow copying TestResult.
  GTEST_DISALLOW_COPY_AND_ASSIGN_(TestResult);
};  // class TestResult

// A TestInfo object stores the following information about a test:
//
//   Test case name
//   Test name
//   Whether the test should be run
//   A function pointer that creates the test object when invoked
//   Test result
//
// The constructor of TestInfo registers itself with the UnitTest
// singleton such that the RUN_ALL_TESTS() macro knows which tests to
// run.
class GTEST_API_ TestInfo {
 public:
  // Destructs a TestInfo object.  This function is not virtual, so
  // don't inherit from TestInfo.
  ~TestInfo();

  // Returns the test case name.
  const char* test_case_name() const { return test_case_name_.c_str(); }

  // Returns the test name.
  const char* name() const { return name_.c_str(); }

  // Returns the name of the parameter type, or NULL if this is not a typed
  // or a type-parameterized test.
  const char* type_param() const {
    if (type_param_.get() != NULL)
      return type_param_->c_str();
    return NULL;
  }

  // Returns the text representation of the value parameter, or NULL if this
  // is not a value-parameterized test.
  const char* value_param() const {
    if (value_param_.get() != NULL)
      return value_param_->c_str();
    return NULL;
  }

  // Returns the file name where this test is defined.
  const char* file() const { return location_.file.c_str(); }

  // Returns the line where this test is defined.
  int line() const { return location_.line; }

  // Return true if this test should not be run because it's in another shard.
  bool is_in_another_shard() const { return is_in_another_shard_; }

  // Returns true if this test should run, that is if the test is not
  // disabled (or it is disabled but the also_run_disabled_tests flag has
  // been specified) and its full name matches the user-specified filter.
  //
  // Google Test allows the user to filter the tests by their full names.
  // The full name of a test Bar in test case Foo is defined as
  // "Foo.Bar".  Only the tests that match the filter will run.
  //
  // A filter is a colon-separated list of glob (not regex) patterns,
  // optionally followed by a '-' and a colon-separated list of
  // negative patterns (tests to exclude).  A test is run if it
  // matches one of the positive patterns and does not match any of
  // the negative patterns.
  //
  // For example, *A*:Foo.* is a filter that matches any string that
  // contains the character 'A' or starts with "Foo.".
  bool should_run() const { return should_run_; }

  // Returns true iff this test will appear in the XML report.
  bool is_reportable() const {
    // The XML report includes tests matching the filter, excluding those
    // run in other shards.
    return matches_filter_ && !is_in_another_shard_;
  }

  // Returns the result of the test.
  const TestResult* result() const { return &result_; }

 private:
#if GTEST_HAS_DEATH_TEST
  friend class internal::DefaultDeathTestFactory;
#endif  // GTEST_HAS_DEATH_TEST
  friend class Test;
  friend class TestCase;
  friend class internal::UnitTestImpl;
  friend class internal::StreamingListenerTest;
  friend 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);

  // Constructs a TestInfo object. The newly constructed instance assumes
  // ownership of the factory object.
  TestInfo(const std::string& test_case_name,
           const std::string& name,
           const char* a_type_param,   // NULL if not a type-parameterized test
           const char* a_value_param,  // NULL if not a value-parameterized test
           internal::CodeLocation a_code_location,
           internal::TypeId fixture_class_id,
           internal::TestFactoryBase* factory);

  // Increments the number of death tests encountered in this test so
  // far.
  int increment_death_test_count() {
    return result_.increment_death_test_count();
  }

  // Creates the test object, runs it, records its result, and then
  // deletes it.
  void Run();

  static void ClearTestResult(TestInfo* test_info) {
    test_info->result_.Clear();
  }

  // These fields are immutable properties of the test.
  const std::string test_case_name_;     // Test case name
  const std::string name_;               // Test name
  // Name of the parameter type, or NULL if this is not a typed or a
  // type-parameterized test.
  const internal::scoped_ptr<const ::std::string> type_param_;
  // Text representation of the value parameter, or NULL if this is not a
  // value-parameterized test.
  const internal::scoped_ptr<const ::std::string> value_param_;
  internal::CodeLocation location_;
  const internal::TypeId fixture_class_id_;   // ID of the test fixture class
  bool should_run_;                 // True iff this test should run
  bool is_disabled_;                // True iff this test is disabled
  bool matches_filter_;             // True if this test matches the
                                    // user-specified filter.
  bool is_in_another_shard_;        // Will be run in another shard.
  internal::TestFactoryBase* const factory_;  // The factory that creates
                                              // the test object

  // This field is mutable and needs to be reset before running the
  // test for the second time.
  TestResult result_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(TestInfo);
};

// A test case, which consists of a vector of TestInfos.
//
// TestCase is not copyable.
class GTEST_API_ TestCase {
 public:
  // Creates a TestCase with the given name.
  //
  // TestCase does NOT have a default constructor.  Always use this
  // constructor to create a TestCase object.
  //
  // Arguments:
  //
  //   name:         name of the test case
  //   a_type_param: the name of the test's type parameter, or NULL if
  //                 this is not a type-parameterized test.
  //   set_up_tc:    pointer to the function that sets up the test case
  //   tear_down_tc: pointer to the function that tears down the test case
  TestCase(const char* name, const char* a_type_param,
           Test::SetUpTestCaseFunc set_up_tc,
           Test::TearDownTestCaseFunc tear_down_tc);

  // Destructor of TestCase.
  virtual ~TestCase();

  // Gets the name of the TestCase.
  const char* name() const { return name_.c_str(); }

  // Returns the name of the parameter type, or NULL if this is not a
  // type-parameterized test case.
  const char* type_param() const {
    if (type_param_.get() != NULL)
      return type_param_->c_str();
    return NULL;
  }

  // Returns true if any test in this test case should run.
  bool should_run() const { return should_run_; }

  // Gets the number of successful tests in this test case.
  int successful_test_count() const;

  // Gets the number of failed tests in this test case.
  int failed_test_count() const;

  // Gets the number of disabled tests that will be reported in the XML report.
  int reportable_disabled_test_count() const;

  // Gets the number of disabled tests in this test case.
  int disabled_test_count() const;

  // Gets the number of tests to be printed in the XML report.
  int reportable_test_count() const;

  // Get the number of tests in this test case that should run.
  int test_to_run_count() const;

  // Gets the number of all tests in this test case.
  int total_test_count() const;

  // Returns true iff the test case passed.
  bool Passed() const { return !Failed(); }

  // Returns true iff the test case failed.
  bool Failed() const { return failed_test_count() > 0; }

  // Returns the elapsed time, in milliseconds.
  TimeInMillis elapsed_time() const { return elapsed_time_; }

  // Returns the i-th test among all the tests. i can range from 0 to
  // total_test_count() - 1. If i is not in that range, returns NULL.
  const TestInfo* GetTestInfo(int i) const;

  // Returns the TestResult that holds test properties recorded during
  // execution of SetUpTestCase and TearDownTestCase.
  const TestResult& ad_hoc_test_result() const { return ad_hoc_test_result_; }

 private:
  friend class Test;
  friend class internal::UnitTestImpl;

  // Gets the (mutable) vector of TestInfos in this TestCase.
  std::vector<TestInfo*>& test_info_list() { return test_info_list_; }

  // Gets the (immutable) vector of TestInfos in this TestCase.
  const std::vector<TestInfo*>& test_info_list() const {
    return test_info_list_;
  }

  // Returns the i-th test among all the tests. i can range from 0 to
  // total_test_count() - 1. If i is not in that range, returns NULL.
  TestInfo* GetMutableTestInfo(int i);

  // Sets the should_run member.
  void set_should_run(bool should) { should_run_ = should; }

  // Adds a TestInfo to this test case.  Will delete the TestInfo upon
  // destruction of the TestCase object.
  void AddTestInfo(TestInfo * test_info);

  // Clears the results of all tests in this test case.
  void ClearResult();

  // Clears the results of all tests in the given test case.
  static void ClearTestCaseResult(TestCase* test_case) {
    test_case->ClearResult();
  }

  // Runs every test in this TestCase.
  void Run();

  // Runs SetUpTestCase() for this TestCase.  This wrapper is needed
  // for catching exceptions thrown from SetUpTestCase().
  void RunSetUpTestCase() { (*set_up_tc_)(); }

  // Runs TearDownTestCase() for this TestCase.  This wrapper is
  // needed for catching exceptions thrown from TearDownTestCase().
  void RunTearDownTestCase() { (*tear_down_tc_)(); }

  // Returns true iff test passed.
  static bool TestPassed(const TestInfo* test_info) {
    return test_info->should_run() && test_info->result()->Passed();
  }

  // Returns true iff test failed.
  static bool TestFailed(const TestInfo* test_info) {
    return test_info->should_run() && test_info->result()->Failed();
  }

  // Returns true iff the test is disabled and will be reported in the XML
  // report.
  static bool TestReportableDisabled(const TestInfo* test_info) {
    return test_info->is_reportable() && test_info->is_disabled_;
  }

  // Returns true iff test is disabled.
  static bool TestDisabled(const TestInfo* test_info) {
    return test_info->is_disabled_;
  }

  // Returns true iff this test will appear in the XML report.
  static bool TestReportable(const TestInfo* test_info) {
    return test_info->is_reportable();
  }

  // Returns true if the given test should run.
  static bool ShouldRunTest(const TestInfo* test_info) {
    return test_info->should_run();
  }

  // Shuffles the tests in this test case.
  void ShuffleTests(internal::Random* random);

  // Restores the test order to before the first shuffle.
  void UnshuffleTests();

  // Name of the test case.
  std::string name_;
  // Name of the parameter type, or NULL if this is not a typed or a
  // type-parameterized test.
  const internal::scoped_ptr<const ::std::string> type_param_;
  // The vector of TestInfos in their original order.  It owns the
  // elements in the vector.
  std::vector<TestInfo*> test_info_list_;
  // Provides a level of indirection for the test list to allow easy
  // shuffling and restoring the test order.  The i-th element in this
  // vector is the index of the i-th test in the shuffled test list.
  std::vector<int> test_indices_;
  // Pointer to the function that sets up the test case.
  Test::SetUpTestCaseFunc set_up_tc_;
  // Pointer to the function that tears down the test case.
  Test::TearDownTestCaseFunc tear_down_tc_;
  // True iff any test in this test case should run.
  bool should_run_;
  // Elapsed time, in milliseconds.
  TimeInMillis elapsed_time_;
  // Holds test properties recorded during execution of SetUpTestCase and
  // TearDownTestCase.
  TestResult ad_hoc_test_result_;

  // We disallow copying TestCases.
  GTEST_DISALLOW_COPY_AND_ASSIGN_(TestCase);
};

// An Environment object is capable of setting up and tearing down an
// environment.  You should subclass this to define your own
// environment(s).
//
// An Environment object does the set-up and tear-down in virtual
// methods SetUp() and TearDown() instead of the constructor and the
// destructor, as:
//
//   1. You cannot safely throw from a destructor.  This is a problem
//      as in some cases Google Test is used where exceptions are enabled, and
//      we may want to implement ASSERT_* using exceptions where they are
//      available.
//   2. You cannot use ASSERT_* directly in a constructor or
//      destructor.
class Environment {
 public:
  // The d'tor is virtual as we need to subclass Environment.
  virtual ~Environment() {}

  // Override this to define how to set up the environment.
  virtual void SetUp() {}

  // Override this to define how to tear down the environment.
  virtual void TearDown() {}
 private:
  // If you see an error about overriding the following function or
  // about it being private, you have mis-spelled SetUp() as Setup().
  struct Setup_should_be_spelled_SetUp {};
  virtual Setup_should_be_spelled_SetUp* Setup() { return NULL; }
};

#if GTEST_HAS_EXCEPTIONS

// Exception which can be thrown from TestEventListener::OnTestPartResult.
class GTEST_API_ AssertionException
    : public internal::GoogleTestFailureException {
 public:
  explicit AssertionException(const TestPartResult& result)
      : GoogleTestFailureException(result) {}
};

#endif  // GTEST_HAS_EXCEPTIONS

// The interface for tracing execution of tests. The methods are organized in
// the order the corresponding events are fired.
class TestEventListener {
 public:
  virtual ~TestEventListener() {}

  // Fired before any test activity starts.
  virtual void OnTestProgramStart(const UnitTest& unit_test) = 0;

  // Fired before each iteration of tests starts.  There may be more than
  // one iteration if GTEST_FLAG(repeat) is set. iteration is the iteration
  // index, starting from 0.
  virtual void OnTestIterationStart(const UnitTest& unit_test,
                                    int iteration) = 0;

  // Fired before environment set-up for each iteration of tests starts.
  virtual void OnEnvironmentsSetUpStart(const UnitTest& unit_test) = 0;

  // Fired after environment set-up for each iteration of tests ends.
  virtual void OnEnvironmentsSetUpEnd(const UnitTest& unit_test) = 0;

  // Fired before the test case starts.
  virtual void OnTestCaseStart(const TestCase& test_case) = 0;

  // Fired before the test starts.
  virtual void OnTestStart(const TestInfo& test_info) = 0;

  // Fired after a failed assertion or a SUCCEED() invocation.
  // If you want to throw an exception from this function to skip to the next
  // TEST, it must be AssertionException defined above, or inherited from it.
  virtual void OnTestPartResult(const TestPartResult& test_part_result) = 0;

  // Fired after the test ends.
  virtual void OnTestEnd(const TestInfo& test_info) = 0;

  // Fired after the test case ends.
  virtual void OnTestCaseEnd(const TestCase& test_case) = 0;

  // Fired before environment tear-down for each iteration of tests starts.
  virtual void OnEnvironmentsTearDownStart(const UnitTest& unit_test) = 0;

  // Fired after environment tear-down for each iteration of tests ends.
  virtual void OnEnvironmentsTearDownEnd(const UnitTest& unit_test) = 0;

  // Fired after each iteration of tests finishes.
  virtual void OnTestIterationEnd(const UnitTest& unit_test,
                                  int iteration) = 0;

  // Fired after all test activities have ended.
  virtual void OnTestProgramEnd(const UnitTest& unit_test) = 0;
};

// The convenience class for users who need to override just one or two
// methods and are not concerned that a possible change to a signature of
// the methods they override will not be caught during the build.  For
// comments about each method please see the definition of TestEventListener
// above.
class EmptyTestEventListener : public TestEventListener {
 public:
  virtual void OnTestProgramStart(const UnitTest& /*unit_test*/) {}
  virtual void OnTestIterationStart(const UnitTest& /*unit_test*/,
                                   int /*iteration*/) {}
  virtual void OnEnvironmentsSetUpStart(const UnitTest& /*unit_test*/) {}
  virtual void OnEnvironmentsSetUpEnd(const UnitTest& /*unit_test*/) {}
  virtual void OnTestCaseStart(const TestCase& /*test_case*/) {}
  virtual void OnTestStart(const TestInfo& /*test_info*/) {}
  virtual void OnTestPartResult(const TestPartResult& /*test_part_result*/) {}
  virtual void OnTestEnd(const TestInfo& /*test_info*/) {}
  virtual void OnTestCaseEnd(const TestCase& /*test_case*/) {}
  virtual void OnEnvironmentsTearDownStart(const UnitTest& /*unit_test*/) {}
  virtual void OnEnvironmentsTearDownEnd(const UnitTest& /*unit_test*/) {}
  virtual void OnTestIterationEnd(const UnitTest& /*unit_test*/,
                                 int /*iteration*/) {}
  virtual void OnTestProgramEnd(const UnitTest& /*unit_test*/) {}
};

// TestEventListeners lets users add listeners to track events in Google Test.
class GTEST_API_ TestEventListeners {
 public:
  TestEventListeners();
  ~TestEventListeners();

  // Appends an event listener to the end of the list. Google Test assumes
  // the ownership of the listener (i.e. it will delete the listener when
  // the test program finishes).
  void Append(TestEventListener* listener);

  // Removes the given event listener from the list and returns it.  It then
  // becomes the caller’s responsibility to delete the listener. Returns
  // NULL if the listener is not found in the list.
  TestEventListener* Release(TestEventListener* listener);

  // Returns the standard listener responsible for the default console
  // output.  Can be removed from the listeners list to shut down default
  // console output.  Note that removing this object from the listener list
  // with Release transfers its ownership to the caller and makes this
  // function return NULL the next time.
  TestEventListener* default_result_printer() const {
    return default_result_printer_;
  }

  // Returns the standard listener responsible for the default XML output
  // controlled by the --gtest_output=xml flag.  Can be removed from the
  // listeners list by users who want to shut down the default XML output
  // controlled by this flag and substitute it with custom one.  Note that
  // removing this object from the listener list with Release transfers its
  // ownership to the caller and makes this function return NULL the next
  // time.
  TestEventListener* default_xml_generator() const {
    return default_xml_generator_;
  }

 private:
  friend class TestCase;
  friend class TestInfo;
  friend class internal::DefaultGlobalTestPartResultReporter;
  friend class internal::NoExecDeathTest;
  friend class internal::TestEventListenersAccessor;
  friend class internal::UnitTestImpl;

  // Returns repeater that broadcasts the TestEventListener events to all
  // subscribers.
  TestEventListener* repeater();

  // Sets the default_result_printer attribute to the provided listener.
  // The listener is also added to the listener list and previous
  // default_result_printer is removed from it and deleted. The listener can
  // also be NULL in which case it will not be added to the list. Does
  // nothing if the previous and the current listener objects are the same.
  void SetDefaultResultPrinter(TestEventListener* listener);

  // Sets the default_xml_generator attribute to the provided listener.  The
  // listener is also added to the listener list and previous
  // default_xml_generator is removed from it and deleted. The listener can
  // also be NULL in which case it will not be added to the list. Does
  // nothing if the previous and the current listener objects are the same.
  void SetDefaultXmlGenerator(TestEventListener* listener);

  // Controls whether events will be forwarded by the repeater to the
  // listeners in the list.
  bool EventForwardingEnabled() const;
  void SuppressEventForwarding();

  // The actual list of listeners.
  internal::TestEventRepeater* repeater_;
  // Listener responsible for the standard result output.
  TestEventListener* default_result_printer_;
  // Listener responsible for the creation of the XML output file.
  TestEventListener* default_xml_generator_;

  // We disallow copying TestEventListeners.
  GTEST_DISALLOW_COPY_AND_ASSIGN_(TestEventListeners);
};

// A UnitTest consists of a vector of TestCases.
//
// This is a singleton class.  The only instance of UnitTest is
// created when UnitTest::GetInstance() is first called.  This
// instance is never deleted.
//
// UnitTest is not copyable.
//
// This class is thread-safe as long as the methods are called
// according to their specification.
class GTEST_API_ UnitTest {
 public:
  // Gets the singleton UnitTest object.  The first time this method
  // is called, a UnitTest object is constructed and returned.
  // Consecutive calls will return the same object.
  static UnitTest* GetInstance();

  // Runs all tests in this UnitTest object and prints the result.
  // Returns 0 if successful, or 1 otherwise.
  //
  // This method can only be called from the main thread.
  //
  // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
  int Run() GTEST_MUST_USE_RESULT_;

  // Returns the working directory when the first TEST() or TEST_F()
  // was executed.  The UnitTest object owns the string.
  const char* original_working_dir() const;

  // Returns the TestCase object for the test that's currently running,
  // or NULL if no test is running.
  const TestCase* current_test_case() const
      GTEST_LOCK_EXCLUDED_(mutex_);

  // Returns the TestInfo object for the test that's currently running,
  // or NULL if no test is running.
  const TestInfo* current_test_info() const
      GTEST_LOCK_EXCLUDED_(mutex_);

  // Returns the random seed used at the start of the current test run.
  int random_seed() const;

  // Returns the ParameterizedTestCaseRegistry object used to keep track of
  // value-parameterized tests and instantiate and register them.
  //
  // INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
  internal::ParameterizedTestCaseRegistry& parameterized_test_registry()
      GTEST_LOCK_EXCLUDED_(mutex_);

  // Gets the number of successful test cases.
  int successful_test_case_count() const;

  // Gets the number of failed test cases.
  int failed_test_case_count() const;

  // Gets the number of all test cases.
  int total_test_case_count() const;

  // Gets the number of all test cases that contain at least one test
  // that should run.
  int test_case_to_run_count() const;

  // Gets the number of successful tests.
  int successful_test_count() const;

  // Gets the number of failed tests.
  int failed_test_count() const;

  // Gets the number of disabled tests that will be reported in the XML report.
  int reportable_disabled_test_count() const;

  // Gets the number of disabled tests.
  int disabled_test_count() const;

  // Gets the number of tests to be printed in the XML report.
  int reportable_test_count() const;

  // Gets the number of all tests.
  int total_test_count() const;

  // Gets the number of tests that should run.
  int test_to_run_count() const;

  // Gets the time of the test program start, in ms from the start of the
  // UNIX epoch.
  TimeInMillis start_timestamp() const;

  // Gets the elapsed time, in milliseconds.
  TimeInMillis elapsed_time() const;

  // Returns true iff the unit test passed (i.e. all test cases passed).
  bool Passed() const;

  // Returns true iff the unit test failed (i.e. some test case failed
  // or something outside of all tests failed).
  bool Failed() const;

  // Gets the i-th test case among all the test cases. i can range from 0 to
  // total_test_case_count() - 1. If i is not in that range, returns NULL.
  const TestCase* GetTestCase(int i) const;

  // Returns the TestResult containing information on test failures and
  // properties logged outside of individual test cases.
  const TestResult& ad_hoc_test_result() const;

  // Returns the list of event listeners that can be used to track events
  // inside Google Test.
  TestEventListeners& listeners();

 private:
  // Registers and returns a global test environment.  When a test
  // program is run, all global test environments will be set-up in
  // the order they were registered.  After all tests in the program
  // have finished, all global test environments will be torn-down in
  // the *reverse* order they were registered.
  //
  // The UnitTest object takes ownership of the given environment.
  //
  // This method can only be called from the main thread.
  Environment* AddEnvironment(Environment* env);

  // Adds a TestPartResult to the current TestResult object.  All
  // Google Test assertion macros (e.g. ASSERT_TRUE, EXPECT_EQ, etc)
  // eventually call this to report their results.  The user code
  // should use the assertion macros instead of calling this directly.
  void AddTestPartResult(TestPartResult::Type result_type,
                         const char* file_name,
                         int line_number,
                         const std::string& message,
                         const std::string& os_stack_trace)
      GTEST_LOCK_EXCLUDED_(mutex_);

  // Adds a TestProperty to the current TestResult object when invoked from
  // inside a test, to current TestCase's ad_hoc_test_result_ when invoked
  // from SetUpTestCase or TearDownTestCase, or to the global property set
  // when invoked elsewhere.  If the result already contains a property with
  // the same key, the value will be updated.
  void RecordProperty(const std::string& key, const std::string& value);

  // Gets the i-th test case among all the test cases. i can range from 0 to
  // total_test_case_count() - 1. If i is not in that range, returns NULL.
  TestCase* GetMutableTestCase(int i);

  // Accessors for the implementation object.
  internal::UnitTestImpl* impl() { return impl_; }
  const internal::UnitTestImpl* impl() const { return impl_; }

  // These classes and functions are friends as they need to access private
  // members of UnitTest.
  friend class ScopedTrace;
  friend class Test;
  friend class internal::AssertHelper;
  friend class internal::StreamingListenerTest;
  friend class internal::UnitTestRecordPropertyTestHelper;
  friend Environment* AddGlobalTestEnvironment(Environment* env);
  friend internal::UnitTestImpl* internal::GetUnitTestImpl();
  friend void internal::ReportFailureInUnknownLocation(
      TestPartResult::Type result_type,
      const std::string& message);

  // Creates an empty UnitTest.
  UnitTest();

  // D'tor
  virtual ~UnitTest();

  // Pushes a trace defined by SCOPED_TRACE() on to the per-thread
  // Google Test trace stack.
  void PushGTestTrace(const internal::TraceInfo& trace)
      GTEST_LOCK_EXCLUDED_(mutex_);

  // Pops a trace from the per-thread Google Test trace stack.
  void PopGTestTrace()
      GTEST_LOCK_EXCLUDED_(mutex_);

  // Protects mutable state in *impl_.  This is mutable as some const
  // methods need to lock it too.
  mutable internal::Mutex mutex_;

  // Opaque implementation object.  This field is never changed once
  // the object is constructed.  We don't mark it as const here, as
  // doing so will cause a warning in the constructor of UnitTest.
  // Mutable state in *impl_ is protected by mutex_.
  internal::UnitTestImpl* impl_;

  // We disallow copying UnitTest.
  GTEST_DISALLOW_COPY_AND_ASSIGN_(UnitTest);
};

// A convenient wrapper for adding an environment for the test
// program.
//
// You should call this before RUN_ALL_TESTS() is called, probably in
// main().  If you use gtest_main, you need to call this before main()
// starts for it to take effect.  For example, you can define a global
// variable like this:
//
//   testing::Environment* const foo_env =
//       testing::AddGlobalTestEnvironment(new FooEnvironment);
//
// However, we strongly recommend you to write your own main() and
// call AddGlobalTestEnvironment() there, as relying on initialization
// of global variables makes the code harder to read and may cause
// problems when you register multiple environments from different
// translation units and the environments have dependencies among them
// (remember that the compiler doesn't guarantee the order in which
// global variables from different translation units are initialized).
inline Environment* AddGlobalTestEnvironment(Environment* env) {
  return UnitTest::GetInstance()->AddEnvironment(env);
}

// Initializes Google Test.  This must be called before calling
// RUN_ALL_TESTS().  In particular, it parses a command line for the
// flags that Google Test recognizes.  Whenever a Google Test flag is
// seen, it is removed from argv, and *argc is decremented.
//
// No value is returned.  Instead, the Google Test flag variables are
// updated.
//
// Calling the function for the second time has no user-visible effect.
GTEST_API_ void InitGoogleTest(int* argc, char** argv);

// This overloaded version can be used in Windows programs compiled in
// UNICODE mode.
GTEST_API_ void InitGoogleTest(int* argc, wchar_t** argv);

namespace internal {

// Separate the error generating code from the code path to reduce the stack
// frame size of CmpHelperEQ. This helps reduce the overhead of some sanitizers
// when calling EXPECT_* in a tight loop.
template <typename T1, typename T2>
AssertionResult CmpHelperEQFailure(const char* lhs_expression,
                                   const char* rhs_expression,
                                   const T1& lhs, const T2& rhs) {
  return EqFailure(lhs_expression,
                   rhs_expression,
                   FormatForComparisonFailureMessage(lhs, rhs),
                   FormatForComparisonFailureMessage(rhs, lhs),
                   false);
}

// The helper function for {ASSERT|EXPECT}_EQ.
template <typename T1, typename T2>
AssertionResult CmpHelperEQ(const char* lhs_expression,
                            const char* rhs_expression,
                            const T1& lhs,
                            const T2& rhs) {
  if (lhs == rhs) {
    return AssertionSuccess();
  }

  return CmpHelperEQFailure(lhs_expression, rhs_expression, lhs, rhs);
}

```

```
// With this overloaded version, we allow anonymous enums to be used
// in {ASSERT|EXPECT}_EQ when compiled with gcc 4, as anonymous enums
// can be implicitly cast to BiggestInt.
GTEST_API_ AssertionResult CmpHelperEQ(const char* lhs_expression,
                                       const char* rhs_expression,
                                       BiggestInt lhs,
                                       BiggestInt rhs);

// The helper class for {ASSERT|EXPECT}_EQ.  The template argument
// lhs_is_null_literal is true iff the first argument to ASSERT_EQ()
// is a null pointer literal.  The following default implementation is
// for lhs_is_null_literal being false.
template <bool lhs_is_null_literal>
class EqHelper {
 public:
  // This templatized version is for the general case.
  template <typename T1, typename T2>
  static AssertionResult Compare(const char* lhs_expression,
                                 const char* rhs_expression,
                                 const T1& lhs,
                                 const T2& rhs) {
    return CmpHelperEQ(lhs_expression, rhs_expression, lhs, rhs);
  }

  // With this overloaded version, we allow anonymous enums to be used
  // in {ASSERT|EXPECT}_EQ when compiled with gcc 4, as anonymous
  // enums can be implicitly cast to BiggestInt.
  //
  // Even though its body looks the same as the above version, we
  // cannot merge the two, as it will make anonymous enums unhappy.
  static AssertionResult Compare(const char* lhs_expression,
                                 const char* rhs_expression,
                                 BiggestInt lhs,
                                 BiggestInt rhs) {
    return CmpHelperEQ(lhs_expression, rhs_expression, lhs, rhs);
  }
};

// This specialization is used when the first argument to ASSERT_EQ()
// is a null pointer literal, like NULL, false, or 0.
template <>
class EqHelper<true> {
 public:
  // We define two overloaded versions of Compare().  The first
  // version will be picked when the second argument to ASSERT_EQ() is
  // NOT a pointer, e.g. ASSERT_EQ(0, AnIntFunction()) or
  // EXPECT_EQ(false, a_bool).
  template <typename T1, typename T2>
  static AssertionResult Compare(
      const char* lhs_expression,
      const char* rhs_expression,
      const T1& lhs,
      const T2& rhs,
      // The following line prevents this overload from being considered if T2
      // is not a pointer type.  We need this because ASSERT_EQ(NULL, my_ptr)
      // expands to Compare("", "", NULL, my_ptr), which requires a conversion
      // to match the Secret* in the other overload, which would otherwise make
      // this template match better.
      typename EnableIf<!is_pointer<T2>::value>::type* = 0) {
    return CmpHelperEQ(lhs_expression, rhs_expression, lhs, rhs);
  }

  // This version will be picked when the second argument to ASSERT_EQ() is a
  // pointer, e.g. ASSERT_EQ(NULL, a_pointer).
  template <typename T>
  static AssertionResult Compare(
      const char* lhs_expression,
      const char* rhs_expression,
      // We used to have a second template parameter instead of Secret*.  That
      // template parameter would deduce to 'long', making this a better match
      // than the first overload even without the first overload's EnableIf.
      // Unfortunately, gcc with -Wconversion-null warns when "passing NULL to
      // non-pointer argument" (even a deduced integral argument), so the old
      // implementation caused warnings in user code.
      Secret* /* lhs (NULL) */,
      T* rhs) {
    // We already know that 'lhs' is a null pointer.
    return CmpHelperEQ(lhs_expression, rhs_expression,
                       static_cast<T*>(NULL), rhs);
  }
};

// Separate the error generating code from the code path to reduce the stack
// frame size of CmpHelperOP. This helps reduce the overhead of some sanitizers
// when calling EXPECT_OP in a tight loop.
template <typename T1, typename T2>
AssertionResult CmpHelperOpFailure(const char* expr1, const char* expr2,
                                   const T1& val1, const T2& val2,
                                   const char* op) {
  return AssertionFailure()
         « "Expected: (" « expr1 « ") " « op « " (" « expr2
         « "), actual: " « FormatForComparisonFailureMessage(val1, val2)
         « " vs " « FormatForComparisonFailureMessage(val2, val1);
}

// A macro for implementing the helper functions needed to implement
// ASSERT_?? and EXPECT_??.  It is here just to avoid copy-and-paste
// of similar code.
//
// For each templatized helper function, we also define an overloaded
// version for BiggestInt in order to reduce code bloat and allow
// anonymous enums to be used with {ASSERT|EXPECT}_?? when compiled
// with gcc 4.
//
// INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.

#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)

// INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.

// Implements the helper function for {ASSERT|EXPECT}_NE
GTEST_IMPL_CMP_HELPER_(NE, !=);
// Implements the helper function for {ASSERT|EXPECT}_LE
GTEST_IMPL_CMP_HELPER_(LE, <=);
// Implements the helper function for {ASSERT|EXPECT}_LT
GTEST_IMPL_CMP_HELPER_(LT, <);
// Implements the helper function for {ASSERT|EXPECT}_GE
GTEST_IMPL_CMP_HELPER_(GE, >=);
// Implements the helper function for {ASSERT|EXPECT}_GT
GTEST_IMPL_CMP_HELPER_(GT, >);

#undef GTEST_IMPL_CMP_HELPER_

// The helper function for {ASSERT|EXPECT}_STREQ.
//
// INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
GTEST_API_ AssertionResult CmpHelperSTREQ(const char* s1_expression,
                                          const char* s2_expression,
                                          const char* s1,
                                          const char* s2);

// The helper function for {ASSERT|EXPECT}_STRCASEEQ.
//
// INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
GTEST_API_ AssertionResult CmpHelperSTRCASEEQ(const char* s1_expression,
                                              const char* s2_expression,
                                              const char* s1,
                                              const char* s2);

// The helper function for {ASSERT|EXPECT}_STRNE.
//
// INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
GTEST_API_ AssertionResult CmpHelperSTRNE(const char* s1_expression,
                                          const char* s2_expression,
                                          const char* s1,
                                          const char* s2);

// The helper function for {ASSERT|EXPECT}_STRCASENE.
//
// INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
GTEST_API_ AssertionResult CmpHelperSTRCASENE(const char* s1_expression,
                                              const char* s2_expression,
                                              const char* s1,
                                              const char* s2);


// Helper function for *_STREQ on wide strings.
//
// INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
GTEST_API_ AssertionResult CmpHelperSTREQ(const char* s1_expression,
                                          const char* s2_expression,
                                          const wchar_t* s1,
                                          const wchar_t* s2);
```

```

// Helper function for *_STRNE on wide strings.
//
// INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
GTEST_API_ AssertionResult CmpHelperSTRNE(const char* s1_expression,
                                          const char* s2_expression,
                                          const wchar_t* s1,
                                          const wchar_t* s2);

}  // namespace internal

// IsSubstring() and IsNotSubstring() are intended to be used as the
// first argument to {EXPECT,ASSERT}_PRED_FORMAT2(), not by
// themselves.  They check whether needle is a substring of haystack
// (NULL is considered a substring of itself only), and return an
// appropriate error message when they fail.
//
// The {needle,haystack}_expr arguments are the stringified
// expressions that generated the two real arguments.
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);

#if GTEST_HAS_STD_WSTRING
GTEST_API_ AssertionResult IsSubstring(
    const char* needle_expr, const char* haystack_expr,
    const ::std::wstring& needle, const ::std::wstring& haystack);
GTEST_API_ AssertionResult IsNotSubstring(
    const char* needle_expr, const char* haystack_expr,
    const ::std::wstring& needle, const ::std::wstring& haystack);
#endif  // GTEST_HAS_STD_WSTRING

namespace internal {

// Helper template function for comparing floating-points.
//
// Template parameter:
//
//   RawType: the raw floating-point type (either float or double)
//
// INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
template <typename RawType>
AssertionResult CmpHelperFloatingPointEQ(const char* lhs_expression,
                                         const char* rhs_expression,
                                         RawType lhs_value,
                                         RawType rhs_value) {
  const FloatingPoint<RawType> lhs(lhs_value), rhs(rhs_value);

  if (lhs.AlmostEquals(rhs)) {
    return AssertionSuccess();
  }

  ::std::stringstream lhs_ss;
  lhs_ss « std::setprecision(std::numeric_limits<RawType>::digits10 + 2)
         « lhs_value;

  ::std::stringstream rhs_ss;
  rhs_ss « std::setprecision(std::numeric_limits<RawType>::digits10 + 2)
         « rhs_value;

  return EqFailure(lhs_expression,
                   rhs_expression,
                   StringStreamToString(&lhs_ss),
                   StringStreamToString(&rhs_ss),
                   false);
}

// Helper function for implementing ASSERT_NEAR.
//
// INTERNAL IMPLEMENTATION - DO NOT USE IN A USER PROGRAM.
GTEST_API_ AssertionResult DoubleNearPredFormat(const char* expr1,
                                                const char* expr2,
                                               const char* abs_error_expr,
                                               double val1,
                                               double val2,
                                               double abs_error);

// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
// A class that enables one to stream messages to assertion macros
class GTEST_API_ AssertHelper {
 public:
  // Constructor.
  AssertHelper(TestPartResult::Type type,
               const char* file,
               int line,
               const char* message);
  ~AssertHelper();

  // Message assignment is a semantic trick to enable assertion
  // streaming; see the GTEST_MESSAGE_ macro below.
  void operator=(const Message& message) const;

 private:
  // We put our data in a struct so that the size of the AssertHelper class can
  // be as small as possible.  This is important because gcc is incapable of
  // re-using stack space even for temporary variables, so every EXPECT_EQ
  // reserves stack space for another AssertHelper.
  struct AssertHelperData {
    AssertHelperData(TestPartResult::Type t,
                     const char* srcfile,
                     int line_num,
                     const char* msg)
        : type(t), file(srcfile), line(line_num), message(msg) { }

    TestPartResult::Type const type;
    const char* const file;
    int const line;
    std::string const message;

   private:
    GTEST_DISALLOW_COPY_AND_ASSIGN_(AssertHelperData);
  };

  AssertHelperData* const data_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(AssertHelper);
};

}  // namespace internal

// The pure interface class that all value-parameterized tests inherit from.
// A value-parameterized class must inherit from both ::testing::Test and
// ::testing::WithParamInterface. In most cases that just means inheriting
// from ::testing::TestWithParam, but more complicated test hierarchies
// may need to inherit from Test and WithParamInterface at different levels.
//
// This interface has support for accessing the test parameter value via
// the GetParam() method.
//
// Use it with one of the parameter generator defining functions, like Range(),
// Values(), ValuesIn(), Bool(), and Combine().
//
// class FooTest : public ::testing::TestWithParam<int> {
//  protected:
//   FooTest() {
//     // Can use GetParam() here.
//   }
//   virtual ~FooTest() {
//     // Can use GetParam() here.
//   }
//   virtual void SetUp() {
//     // Can use GetParam() here.
//   }
//   virtual void TearDown {
//     // Can use GetParam() here.
//   }
// };
// TEST_P(FooTest, DoesBar) {
//   // Can use GetParam() method here.
//   Foo foo;
//   ASSERT_TRUE(foo.DoesBar(GetParam()));
// }
// INSTANTIATE_TEST_CASE_P(OneToTenRange, FooTest, ::testing::Range(1, 10));

template <typename T>
class WithParamInterface {
 public:
  typedef T ParamType;
  virtual ~WithParamInterface() {}

  // The current parameter value. Is also available in the test fixture's
  // constructor. This member function is non-static, even though it only
  // references static data, to reduce the opportunity for incorrect uses
  // like writing 'WithParamInterface<bool>::GetParam()' for a test that
  // uses a fixture whose parameter type is int.
  const ParamType& GetParam() const {
    GTEST_CHECK_(parameter_ != NULL)
        « "GetParam() can only be called inside a value-parameterized test "
        « "-- did you intend to write TEST_P instead of TEST_F?";
    return *parameter_;
  }

 private:
  // Sets parameter value. The caller is responsible for making sure the value
  // remains alive and unchanged throughout the current test.
  static void SetParam(const ParamType* parameter) {
    parameter_ = parameter;
  }

  // Static value used for accessing parameter during a test lifetime.
  static const ParamType* parameter_;

  // TestClass must be a subclass of WithParamInterface<T> and Test.
  template <class TestClass> friend class internal::ParameterizedTestFactory;
};

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

// Most value-parameterized classes can ignore the existence of
// WithParamInterface, and can just inherit from ::testing::TestWithParam.

template <typename T>
class TestWithParam : public Test, public WithParamInterface<T> {
};

// Macros for indicating success/failure in test code.

// ADD_FAILURE unconditionally adds a failure to the current test.
// SUCCEED generates a success - it doesn't automatically make the
// current test successful, as a test is only successful when it has
// no failure.
//
// EXPECT_* verifies that a certain condition is satisfied.  If not,
// it behaves like ADD_FAILURE.  In particular:
//
//   EXPECT_TRUE  verifies that a Boolean condition is true.
//   EXPECT_FALSE verifies that a Boolean condition is false.
//
// FAIL and ASSERT_* are similar to ADD_FAILURE and EXPECT_*, except
// that they will also abort the current function on failure.  People
// usually want the fail-fast behavior of FAIL and ASSERT_*, but those
// writing data-driven tests often find themselves using ADD_FAILURE
// and EXPECT_* more.

// Generates a nonfatal failure with a generic message.
#define ADD_FAILURE() GTEST_NONFATAL_FAILURE_("Failed")

// Generates a nonfatal failure at the given source file location with
// a generic message.
#define ADD_FAILURE_AT(file, line) \
  GTEST_MESSAGE_AT_(file, line, "Failed", \
                    ::testing::TestPartResult::kNonFatalFailure)

// Generates a fatal failure with a generic message.
#define GTEST_FAIL() GTEST_FATAL_FAILURE_("Failed")

// Define this macro to 1 to omit the definition of FAIL(), which is a
// generic name and clashes with some other libraries.
#if !GTEST_DONT_DEFINE_FAIL
# define FAIL() GTEST_FAIL()
#endif

// Generates a success with a generic message.
#define GTEST_SUCCEED() GTEST_SUCCESS_("Succeeded")

// Define this macro to 1 to omit the definition of SUCCEED(), which
// is a generic name and clashes with some other libraries.
#if !GTEST_DONT_DEFINE_SUCCEED
# define SUCCEED() GTEST_SUCCEED()
#endif

// Macros for testing exceptions.
//
//    * {ASSERT|EXPECT}_THROW(statement, expected_exception):
//         Tests that the statement throws the expected exception.
//    * {ASSERT|EXPECT}_NO_THROW(statement):
//         Tests that the statement doesn't throw any exception.
//    * {ASSERT|EXPECT}_ANY_THROW(statement):
//         Tests that the statement throws an exception.

#define EXPECT_THROW(statement, expected_exception) \
  GTEST_TEST_THROW_(statement, expected_exception, GTEST_NONFATAL_FAILURE_)
#define EXPECT_NO_THROW(statement) \
  GTEST_TEST_NO_THROW_(statement, GTEST_NONFATAL_FAILURE_)
#define EXPECT_ANY_THROW(statement) \
  GTEST_TEST_ANY_THROW_(statement, GTEST_NONFATAL_FAILURE_)
#define ASSERT_THROW(statement, expected_exception) \
  GTEST_TEST_THROW_(statement, expected_exception, GTEST_FATAL_FAILURE_)
#define ASSERT_NO_THROW(statement) \
  GTEST_TEST_NO_THROW_(statement, GTEST_FATAL_FAILURE_)
#define ASSERT_ANY_THROW(statement) \
  GTEST_TEST_ANY_THROW_(statement, GTEST_FATAL_FAILURE_)

// Boolean assertions. Condition can be either a Boolean expression or an
// AssertionResult. For more information on how to use AssertionResult with
// these macros see comments on that class.
#define EXPECT_TRUE(condition) \
  GTEST_TEST_BOOLEAN_(condition, #condition, false, true, \
                      GTEST_NONFATAL_FAILURE_)
#define EXPECT_FALSE(condition) \
  GTEST_TEST_BOOLEAN_(!(condition), #condition, true, false, \
                      GTEST_NONFATAL_FAILURE_)
#define ASSERT_TRUE(condition) \
  GTEST_TEST_BOOLEAN_(condition, #condition, false, true, \
                      GTEST_FATAL_FAILURE_)
#define ASSERT_FALSE(condition) \
  GTEST_TEST_BOOLEAN_(!(condition), #condition, true, false, \
                      GTEST_FATAL_FAILURE_)

// Macros for testing equalities and inequalities.
//
//    * {ASSERT|EXPECT}_EQ(v1, v2): Tests that v1 == v2
//    * {ASSERT|EXPECT}_NE(v1, v2): Tests that v1 != v2
//    * {ASSERT|EXPECT}_LT(v1, v2): Tests that v1 < v2
//    * {ASSERT|EXPECT}_LE(v1, v2): Tests that v1 <= v2
//    * {ASSERT|EXPECT}_GT(v1, v2): Tests that v1 > v2
//    * {ASSERT|EXPECT}_GE(v1, v2): Tests that v1 >= v2
//
// When they are not, Google Test prints both the tested expressions and
// their actual values.  The values must be compatible built-in types,
// or you will get a compiler error.  By "compatible" we mean that the
// values can be compared by the respective operator.
//
// Note:
//
//   1. It is possible to make a user-defined type work with
//   {ASSERT|EXPECT}_??(), but that requires overloading the
//   comparison operators and is thus discouraged by the Google C++
//   Usage Guide.  Therefore, you are advised to use the
//   {ASSERT|EXPECT}_TRUE() macro to assert that two objects are
//   equal.
//
//   2. The {ASSERT|EXPECT}_??() macros do pointer comparisons on
//   pointers (in particular, C strings).  Therefore, if you use it
//   with two C strings, you are testing how their locations in memory
//   are related, not how their content is related.  To compare two C
//   strings by content, use {ASSERT|EXPECT}_STR*().
//
//   3. {ASSERT|EXPECT}_EQ(v1, v2) is preferred to
//   {ASSERT|EXPECT}_TRUE(v1 == v2), as the former tells you
//   what the actual value is when it fails, and similarly for the
//   other comparisons.
//
//   4. Do not depend on the order in which {ASSERT|EXPECT}_??()
//   evaluate their arguments, which is undefined.
//
//   5. These macros evaluate their arguments exactly once.
//
// Examples:
//
//   EXPECT_NE(Foo(), 5);
//   EXPECT_EQ(a_pointer, NULL);
//   ASSERT_LT(i, array_size);
//   ASSERT_GT(records.size(), 0) « "There is no record left.";

#define EXPECT_EQ(val1, val2) \
  EXPECT_PRED_FORMAT2(::testing::internal:: \
                      EqHelper<GTEST_IS_NULL_LITERAL_(val1)>::Compare, \
                      val1, val2)
#define EXPECT_NE(val1, val2) \
  EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperNE, val1, val2)
#define EXPECT_LE(val1, val2) \
  EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperLE, val1, val2)
#define EXPECT_LT(val1, val2) \
  EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperLT, val1, val2)
#define EXPECT_GE(val1, val2) \
  EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperGE, val1, val2)
#define EXPECT_GT(val1, val2) \
  EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperGT, val1, val2)

#define GTEST_ASSERT_EQ(val1, val2) \
  ASSERT_PRED_FORMAT2(::testing::internal:: \
                      EqHelper<GTEST_IS_NULL_LITERAL_(val1)>::Compare, \
                      val1, val2)
#define GTEST_ASSERT_NE(val1, val2) \
  ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperNE, val1, val2)
#define GTEST_ASSERT_LE(val1, val2) \
  ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperLE, val1, val2)
#define GTEST_ASSERT_LT(val1, val2) \
  ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperLT, val1, val2)
#define GTEST_ASSERT_GE(val1, val2) \
  ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperGE, val1, val2)
#define GTEST_ASSERT_GT(val1, val2) \
  ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperGT, val1, val2)

// Define macro GTEST_DONT_DEFINE_ASSERT_XY to 1 to omit the definition of
// ASSERT_XY(), which clashes with some users' own code.

#if !GTEST_DONT_DEFINE_ASSERT_EQ
# define ASSERT_EQ(val1, val2) GTEST_ASSERT_EQ(val1, val2)
#endif

#if !GTEST_DONT_DEFINE_ASSERT_NE
# define ASSERT_NE(val1, val2) GTEST_ASSERT_NE(val1, val2)
#endif

#if !GTEST_DONT_DEFINE_ASSERT_LE
# define ASSERT_LE(val1, val2) GTEST_ASSERT_LE(val1, val2)
#endif

#if !GTEST_DONT_DEFINE_ASSERT_LT
# define ASSERT_LT(val1, val2) GTEST_ASSERT_LT(val1, val2)
#endif

#if !GTEST_DONT_DEFINE_ASSERT_GE
# define ASSERT_GE(val1, val2) GTEST_ASSERT_GE(val1, val2)
#endif

#if !GTEST_DONT_DEFINE_ASSERT_GT
# define ASSERT_GT(val1, val2) GTEST_ASSERT_GT(val1, val2)
#endif

// C-string Comparisons.  All tests treat NULL and any non-NULL string
// as different.  Two NULLs are equal.
//
//    * {ASSERT|EXPECT}_STREQ(s1, s2):     Tests that s1 == s2
//    * {ASSERT|EXPECT}_STRNE(s1, s2):     Tests that s1 != s2
//    * {ASSERT|EXPECT}_STRCASEEQ(s1, s2): Tests that s1 == s2, ignoring case
//    * {ASSERT|EXPECT}_STRCASENE(s1, s2): Tests that s1 != s2, ignoring case
//
// For wide or narrow string objects, you can use the
// {ASSERT|EXPECT}_??() macros.
//
// Don't depend on the order in which the arguments are evaluated,
// which is undefined.
//
// These macros evaluate their arguments exactly once.

#define EXPECT_STREQ(s1, s2) \
  EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperSTREQ, s1, s2)
#define EXPECT_STRNE(s1, s2) \
  EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperSTRNE, s1, s2)
#define EXPECT_STRCASEEQ(s1, s2) \
  EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperSTRCASEEQ, s1, s2)
#define EXPECT_STRCASENE(s1, s2)\
  EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperSTRCASENE, s1, s2)

#define ASSERT_STREQ(s1, s2) \
  ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperSTREQ, s1, s2)
#define ASSERT_STRNE(s1, s2) \
  ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperSTRNE, s1, s2)
#define ASSERT_STRCASEEQ(s1, s2) \
  ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperSTRCASEEQ, s1, s2)
#define ASSERT_STRCASENE(s1, s2)\
  ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperSTRCASENE, s1, s2)

// Macros for comparing floating-point numbers.
//
//    * {ASSERT|EXPECT}_FLOAT_EQ(val1, val2):
```

```
//         Tests that two float values are almost equal.
//    * {ASSERT|EXPECT}_DOUBLE_EQ(val1, val2):
//         Tests that two double values are almost equal.
//    * {ASSERT|EXPECT}_NEAR(v1, v2, abs_error):
//         Tests that v1 and v2 are within the given distance to each other.
//
// Google Test uses ULP-based comparison to automatically pick a default
// error bound that is appropriate for the operands.  See the
// FloatingPoint template class in gtest-internal.h if you are
// interested in the implementation details.

#define EXPECT_FLOAT_EQ(val1, val2)\
  EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperFloatingPointEQ<float>, \
                      val1, val2)

#define EXPECT_DOUBLE_EQ(val1, val2)\
  EXPECT_PRED_FORMAT2(::testing::internal::CmpHelperFloatingPointEQ<double>, \
                      val1, val2)

#define ASSERT_FLOAT_EQ(val1, val2)\
  ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperFloatingPointEQ<float>, \
                      val1, val2)

#define ASSERT_DOUBLE_EQ(val1, val2)\
  ASSERT_PRED_FORMAT2(::testing::internal::CmpHelperFloatingPointEQ<double>, \
                      val1, val2)

#define EXPECT_NEAR(val1, val2, abs_error)\
  EXPECT_PRED_FORMAT3(::testing::internal::DoubleNearPredFormat, \
                      val1, val2, abs_error)

#define ASSERT_NEAR(val1, val2, abs_error)\
  ASSERT_PRED_FORMAT3(::testing::internal::DoubleNearPredFormat, \
                      val1, val2, abs_error)

// These predicate format functions work on floating-point values, and
// can be used in {ASSERT|EXPECT}_PRED_FORMAT2*(), e.g.
//
//   EXPECT_PRED_FORMAT2(testing::DoubleLE, Foo(), 5.0);

// Asserts that val1 is less than, or almost equal to, val2.  Fails
// otherwise.  In particular, it fails if either val1 or val2 is NaN.
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);


#if GTEST_OS_WINDOWS

// Macros that test for HRESULT failure and success, these are only useful
// on Windows, and rely on Windows SDK macros and APIs to compile.
//
//    * {ASSERT|EXPECT}_HRESULT_{SUCCEEDED|FAILED}(expr)
//
// When expr unexpectedly fails or succeeds, Google Test prints the
// expected result and the actual result with both a human-readable
// string representation of the error, if available, as well as the
// hex result code.
# define EXPECT_HRESULT_SUCCEEDED(expr) \
    EXPECT_PRED_FORMAT1(::testing::internal::IsHRESULTSuccess, (expr))

# define ASSERT_HRESULT_SUCCEEDED(expr) \
    ASSERT_PRED_FORMAT1(::testing::internal::IsHRESULTSuccess, (expr))

# define EXPECT_HRESULT_FAILED(expr) \
    EXPECT_PRED_FORMAT1(::testing::internal::IsHRESULTFailure, (expr))

# define ASSERT_HRESULT_FAILED(expr) \
    ASSERT_PRED_FORMAT1(::testing::internal::IsHRESULTFailure, (expr))

#endif  // GTEST_OS_WINDOWS

// Macros that execute statement and check that it doesn't generate new fatal
// failures in the current thread.
//
//   * {ASSERT|EXPECT}_NO_FATAL_FAILURE(statement);
//
// Examples:
//
//   EXPECT_NO_FATAL_FAILURE(Process());
//   ASSERT_NO_FATAL_FAILURE(Process()) « "Process() failed";
//
#define ASSERT_NO_FATAL_FAILURE(statement) \
    GTEST_TEST_NO_FATAL_FAILURE_(statement, GTEST_FATAL_FAILURE_)
#define EXPECT_NO_FATAL_FAILURE(statement) \
    GTEST_TEST_NO_FATAL_FAILURE_(statement, GTEST_NONFATAL_FAILURE_)
```

```

// Causes a trace (including the given source file path and line number,
// and the given message) to be included in every test failure message generated
// by code in the scope of the lifetime of an instance of this class. The effect
// is undone with the destruction of the instance.
//
// The message argument can be anything streamable to std::ostream.
//
// Example:
//   testing::ScopedTrace trace("file.cc", 123, "message");
//
class GTEST_API_ ScopedTrace {
 public:
  // The c'tor pushes the given source file location and message onto
  // a trace stack maintained by Google Test.

  // Template version. Uses Message() to convert the values into strings.
  // Slow, but flexible.
  template <typename T>
  ScopedTrace(const char* file, int line, const T& message) {
    PushTrace(file, line, (Message() « message).GetString());
  }

  // Optimize for some known types.
  ScopedTrace(const char* file, int line, const char* message) {
    PushTrace(file, line, message ? message : "(null)");
  }

#if GTEST_HAS_GLOBAL_STRING
  ScopedTrace(const char* file, int line, const ::string& message) {
    PushTrace(file, line, message);
  }
#endif

  ScopedTrace(const char* file, int line, const std::string& message) {
    PushTrace(file, line, message);
  }

  // The d'tor pops the info pushed by the c'tor.
  //
  // Note that the d'tor is not virtual in order to be efficient.
  // Don't inherit from ScopedTrace!
  ~ScopedTrace();

 private:
  void PushTrace(const char* file, int line, std::string message);

  GTEST_DISALLOW_COPY_AND_ASSIGN_(ScopedTrace);
} GTEST_ATTRIBUTE_UNUSED_;  // A ScopedTrace object does its job in its
                            // c'tor and d'tor.  Therefore it doesn't
                            // need to be used otherwise.

// Causes a trace (including the source file path, the current line
// number, and the given message) to be included in every test failure
// message generated by code in the current scope.  The effect is
// undone when the control leaves the current scope.
//
// The message argument can be anything streamable to std::ostream.
//
// In the implementation, we include the current line number as part
// of the dummy variable name, thus allowing multiple SCOPED_TRACE()s
// to appear in the same block - as long as they are on different
// lines.
//
// Assuming that each thread maintains its own stack of traces.
// Therefore, a SCOPED_TRACE() would (correctly) only affect the
// assertions in its own thread.
#define SCOPED_TRACE(message) \
  ::testing::ScopedTrace GTEST_CONCAT_TOKEN_(gtest_trace_, __LINE__)(\
    __FILE__, __LINE__, (message))


// Compile-time assertion for type equality.
// StaticAssertTypeEq<type1, type2>() compiles iff type1 and type2 are
// the same type.  The value it returns is not interesting.
//
// Instead of making StaticAssertTypeEq a class template, we make it a
// function template that invokes a helper class template.  This
// prevents a user from misusing StaticAssertTypeEq<T1, T2> by
// defining objects of that type.
//
// CAVEAT:
//
// When used inside a method of a class template,
// StaticAssertTypeEq<T1, T2>() is effective ONLY IF the method is
// instantiated.  For example, given:
//
```

```
//   template <typename T> class Foo {
//    public:
//     void Bar() { testing::StaticAssertTypeEq<int, T>(); }
//   };
//
// the code:
//
//   void Test1() { Foo<bool> foo; }
//
// will NOT generate a compiler error, as Foo<bool>::Bar() is never
// actually instantiated.  Instead, you need:
//
//   void Test2() { Foo<bool> foo; foo.Bar(); }
//
// to cause a compiler error.
template <typename T1, typename T2>
bool StaticAssertTypeEq() {
  (void)internal::StaticAssertTypeEqHelper<T1, T2>();
  return true;
}

// Defines a test.
//
// The first parameter is the name of the test case, and the second
// parameter is the name of the test within the test case.
//
// The convention is to end the test case name with "Test".  For
// example, a test case for the Foo class can be named FooTest.
//
// Test code should appear between braces after an invocation of
// this macro.  Example:
//
//   TEST(FooTest, InitializesCorrectly) {
//     Foo foo;
//     EXPECT_TRUE(foo.StatusIsOK());
//   }

// Note that we call GetTestTypeId() instead of GetTypeId<
// ::testing::Test>() here to get the type ID of testing::Test.  This
// is to work around a suspected linker bug when using Google Test as
// a framework on Mac OS X.  The bug causes GetTypeId<
// ::testing::Test>() to return different values depending on whether
// the call is from the Google Test framework itself or from user test
// code.  GetTestTypeId() is guaranteed to always return the same
// value, as it always calls GetTypeId<>() from the Google Test
// framework.
#define GTEST_TEST(test_case_name, test_name)\
  GTEST_TEST_(test_case_name, test_name, \
              ::testing::Test, ::testing::internal::GetTestTypeId())

// Define this macro to 1 to omit the definition of TEST(), which
// is a generic name and clashes with some other libraries.
#if !GTEST_DONT_DEFINE_TEST
# define TEST(test_case_name, test_name) GTEST_TEST(test_case_name, test_name)
#endif

// Defines a test that uses a test fixture.
//
// The first parameter is the name of the test fixture class, which
// also doubles as the test case name.  The second parameter is the
// name of the test within the test case.
//
// A test fixture class must be declared earlier.  The user should put
// the test code between braces after using this macro.  Example:
//
//   class FooTest : public testing::Test {
//    protected:
//     virtual void SetUp() { b_.AddElement(3); }
//
//     Foo a_;
//     Foo b_;
//   };
//
//   TEST_F(FooTest, InitializesCorrectly) {
//     EXPECT_TRUE(a_.StatusIsOK());
//   }
//
//   TEST_F(FooTest, ReturnsElementCountCorrectly) {
//     EXPECT_EQ(a_.size(), 0);
//     EXPECT_EQ(b_.size(), 1);
//   }

#define TEST_F(test_fixture, test_name)\
  GTEST_TEST_(test_fixture, test_name, test_fixture, \
              ::testing::internal::GetTypeId<test_fixture>())

// Returns a path to temporary directory.
// Tries to determine an appropriate directory for the platform.
GTEST_API_ std::string TempDir();

#ifdef _MSC_VER
#  pragma warning(pop)
#endif

}  // namespace testing

// Use this function in main() to run all tests.  It returns 0 if all
// tests are successful, or 1 otherwise.
//
// RUN_ALL_TESTS() should be invoked after the command line has been
// parsed by InitGoogleTest().
//
// This function was formerly a macro; thus, it is in the global
// namespace and has an all-caps name.
int RUN_ALL_TESTS() GTEST_MUST_USE_RESULT_;

inline int RUN_ALL_TESTS() {
  return ::testing::UnitTest::GetInstance()->Run();
}

GTEST_DISABLE_MSC_WARNINGS_POP_()  //  4251

#endif  // GTEST_INCLUDE_GTEST_GTEST_H_
```

9.22 Dokumentacja pliku packages/Microsoft.googletest.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.

```
// Copyright 2015, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// Injection point for custom user configurations. See README for details
//
// ** Custom implementation starts here **

#ifndef GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_H_
#define GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_H_

#endif  // GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_H_
```

9.24 Dokumentacja pliku packages/Microsoft.googletest.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.

```
// Copyright 2006, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

// This file is AUTOMATICALLY GENERATED on 01/02/2018 by command
// 'gen_gtest_pred_impl.py 5'.  DO NOT EDIT BY HAND!
//
// Implements a family of generic predicate assertion macros.

// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_GTEST_PRED_IMPL_H_
#define GTEST_INCLUDE_GTEST_GTEST_PRED_IMPL_H_

#include "gtest/gtest.h"

namespace testing {

// This header implements a family of generic predicate assertion
// macros:
//
//   ASSERT_PRED_FORMAT1(pred_format, v1)
//   ASSERT_PRED_FORMAT2(pred_format, v1, v2)
//   ...
//
// where pred_format is a function or functor that takes n (in the
// case of ASSERT_PRED_FORMATn) values and their source expression
// text, and returns a testing::AssertionResult.  See the definition
// of ASSERT_EQ in gtest.h for an example.
//
// If you don't care about formatting, you can use the more
// restrictive version:
//
//   ASSERT_PRED1(pred, v1)
//   ASSERT_PRED2(pred, v1, v2)
//   ...
//
// where pred is an n-ary function or functor that returns bool,
// and the values v1, v2, ..., must support the « operator for
// streaming to std::ostream.
//
// We also define the EXPECT_* variations.
//
// For now we only support predicates whose arity is at most 5.

// GTEST_ASSERT_ is the basic statement to which all of the assertions
// in this file reduce.  Don't use this in your code.

#define GTEST_ASSERT_(expression, on_failure) \
  GTEST_AMBIGUOUS_ELSE_BLOCKER_ \
  if (const ::testing::AssertionResult gtest_ar = (expression)) \
    ; \
  else \
    on_failure(gtest_ar.failure_message())


// Helper function for implementing {EXPECT|ASSERT}_PRED1.  Don't use
// this in your code.
template <typename Pred,
          typename T1>
AssertionResult AssertPred1Helper(const char* pred_text,
                                  const char* e1,
                                  Pred pred,
                                  const T1& v1) {
  if (pred(v1)) return AssertionSuccess();

  return AssertionFailure() « pred_text « "("
                            « e1 « ") evaluates to false, where"
                            « "\n" « e1 « " evaluates to " « v1;
}

// Internal macro for implementing {EXPECT|ASSERT}_PRED_FORMAT1.
// Don't use this in your code.
#define GTEST_PRED_FORMAT1_(pred_format, v1, on_failure)\
  GTEST_ASSERT_(pred_format(#v1, v1), \
                on_failure)

// Internal macro for implementing {EXPECT|ASSERT}_PRED1.  Don't use
// this in your code.
#define GTEST_PRED1_(pred, v1, on_failure)\
  GTEST_ASSERT_(::testing::AssertPred1Helper(#pred, \
                                             #v1, \
                                             pred, \
                                             v1), on_failure)

// Unary predicate assertion macros.
#define EXPECT_PRED_FORMAT1(pred_format, v1) \
  GTEST_PRED_FORMAT1_(pred_format, v1, GTEST_NONFATAL_FAILURE_)
#define EXPECT_PRED1(pred, v1) \
  GTEST_PRED1_(pred, v1, GTEST_NONFATAL_FAILURE_)
#define ASSERT_PRED_FORMAT1(pred_format, v1) \
  GTEST_PRED_FORMAT1_(pred_format, v1, GTEST_FATAL_FAILURE_)
#define ASSERT_PRED1(pred, v1) \
  GTEST_PRED1_(pred, v1, GTEST_FATAL_FAILURE_)

```

```


// Helper function for implementing {EXPECT|ASSERT}_PRED2.  Don’t use
// this in your code.
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) {
  if (pred(v1, v2)) return AssertionSuccess();

  return AssertionFailure() « pred_text « "("
                            « e1 « ", "
                            « e2 « ") evaluates to false, where"
                            « "\n" « e1 « " evaluates to " « v1
                            « "\n" « e2 « " evaluates to " « v2;
}

// Internal macro for implementing {EXPECT|ASSERT}_PRED_FORMAT2.
// Don’t use this in your code.
#define GTEST_PRED_FORMAT2_(pred_format, v1, v2, on_failure)\
  GTEST_ASSERT_(pred_format(#v1, #v2, v1, v2), \
                on_failure)

// Internal macro for implementing {EXPECT|ASSERT}_PRED2.  Don’t use
// this in your code.
#define GTEST_PRED2_(pred, v1, v2, on_failure)\
  GTEST_ASSERT_(::testing::AssertPred2Helper(#pred, \
                                             #v1, \
                                             #v2, \
                                             pred, \
                                             v1, \
                                             v2), on_failure)

// Binary predicate assertion macros.
#define EXPECT_PRED_FORMAT2(pred_format, v1, v2) \
  GTEST_PRED_FORMAT2_(pred_format, v1, v2, GTEST_NONFATAL_FAILURE_)
#define EXPECT_PRED2(pred, v1, v2) \
  GTEST_PRED2_(pred, v1, v2, GTEST_NONFATAL_FAILURE_)
#define ASSERT_PRED_FORMAT2(pred_format, v1, v2) \
  GTEST_PRED_FORMAT2_(pred_format, v1, v2, GTEST_FATAL_FAILURE_)
#define ASSERT_PRED2(pred, v1, v2) \
  GTEST_PRED2_(pred, v1, v2, GTEST_FATAL_FAILURE_)



// Helper function for implementing {EXPECT|ASSERT}_PRED3.  Don’t use
// this in your code.
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) {
  if (pred(v1, v2, v3)) return AssertionSuccess();

  return AssertionFailure() « pred_text « "("
                            « e1 « ", "
                            « e2 « ", "
                            « e3 « ") evaluates to false, where"
                            « "\n" « e1 « " evaluates to " « v1
                            « "\n" « e2 « " evaluates to " « v2
                            « "\n" « e3 « " evaluates to " « v3;
}

// Internal macro for implementing {EXPECT|ASSERT}_PRED_FORMAT3.
// Don’t use this in your code.
#define GTEST_PRED_FORMAT3_(pred_format, v1, v2, v3, on_failure)\
  GTEST_ASSERT_(pred_format(#v1, #v2, #v3, v1, v2, v3), \
                on_failure)

// Internal macro for implementing {EXPECT|ASSERT}_PRED3.  Don’t use
// this in your code.
#define GTEST_PRED3_(pred, v1, v2, v3, on_failure)\
  GTEST_ASSERT_(::testing::AssertPred3Helper(#pred, \
                                             #v1, \
                                             #v2, \
                                             #v3, \
                                             pred, \
                                             v1, \
                                             v2, \
                                             v3), on_failure)

// Ternary predicate assertion macros.
#define EXPECT_PRED_FORMAT3(pred_format, v1, v2, v3) \
  GTEST_PRED_FORMAT3_(pred_format, v1, v2, v3, GTEST_NONFATAL_FAILURE_)
#define EXPECT_PRED3(pred, v1, v2, v3) \
  GTEST_PRED3_(pred, v1, v2, v3, GTEST_NONFATAL_FAILURE_)
#define ASSERT_PRED_FORMAT3(pred_format, v1, v2, v3) \
  GTEST_PRED_FORMAT3_(pred_format, v1, v2, v3, GTEST_FATAL_FAILURE_)
#define ASSERT_PRED3(pred, v1, v2, v3) \
  GTEST_PRED3_(pred, v1, v2, v3, GTEST_FATAL_FAILURE_)



// Helper function for implementing {EXPECT|ASSERT}_PRED4.  Don’t use
// this in your code.
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) {
  if (pred(v1, v2, v3, v4)) return AssertionSuccess();

  return AssertionFailure() « pred_text « "("
                            « e1 « ", "
                            « e2 « ", "
                            « e3 « ", "
                            « e4 « ") evaluates to false, where"
                            « "\n" « e1 « " evaluates to " « v1
                            « "\n" « e2 « " evaluates to " « v2
                            « "\n" « e3 « " evaluates to " « v3
                            « "\n" « e4 « " evaluates to " « v4;
}

// Internal macro for implementing {EXPECT|ASSERT}_PRED_FORMAT4.
// Don’t use this in your code.
#define GTEST_PRED_FORMAT4_(pred_format, v1, v2, v3, v4, on_failure)\
  GTEST_ASSERT_(pred_format(#v1, #v2, #v3, #v4, v1, v2, v3, v4), \
                on_failure)

// Internal macro for implementing {EXPECT|ASSERT}_PRED4.  Don’t use
// this in your code.
#define GTEST_PRED4_(pred, v1, v2, v3, v4, on_failure)\
  GTEST_ASSERT_(::testing::AssertPred4Helper(#pred, \
                                             #v1, \
                                             #v2, \
                                             #v3, \
                                             #v4, \
                                             pred, \
                                             v1, \
                                             v2, \
                                             v3, \
                                             v4), on_failure)

// 4-ary predicate assertion macros.
#define EXPECT_PRED_FORMAT4(pred_format, v1, v2, v3, v4) \
  GTEST_PRED_FORMAT4_(pred_format, v1, v2, v3, v4, GTEST_NONFATAL_FAILURE_)
#define EXPECT_PRED4(pred, v1, v2, v3, v4) \
  GTEST_PRED4_(pred, v1, v2, v3, v4, GTEST_NONFATAL_FAILURE_)
#define ASSERT_PRED_FORMAT4(pred_format, v1, v2, v3, v4) \
  GTEST_PRED_FORMAT4_(pred_format, v1, v2, v3, v4, GTEST_FATAL_FAILURE_)
#define ASSERT_PRED4(pred, v1, v2, v3, v4) \
  GTEST_PRED4_(pred, v1, v2, v3, v4, GTEST_FATAL_FAILURE_)



// Helper function for implementing {EXPECT|ASSERT}_PRED5.  Don’t use
// this in your code.
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) {
  if (pred(v1, v2, v3, v4, v5)) return AssertionSuccess();

  return AssertionFailure() « pred_text « "("
                            « e1 « ", "
                            « e2 « ", "
                            « e3 « ", "
                            « e4 « ", "
                            « e5 « ") evaluates to false, where"
                            « "\n" « e1 « " evaluates to " « v1
                            « "\n" « e2 « " evaluates to " « v2
                            « "\n" « e3 « " evaluates to " « v3
                            « "\n" « e4 « " evaluates to " « v4
                            « "\n" « e5 « " evaluates to " « v5;
}

// Internal macro for implementing {EXPECT|ASSERT}_PRED_FORMAT5.
// Don't use this in your code.
#define GTEST_PRED_FORMAT5_(pred_format, v1, v2, v3, v4, v5, on_failure)\
  GTEST_ASSERT_(pred_format(#v1, #v2, #v3, #v4, #v5, v1, v2, v3, v4, v5), \
                on_failure)

// Internal macro for implementing {EXPECT|ASSERT}_PRED5.  Don't use
// this in your code.
#define GTEST_PRED5_(pred, v1, v2, v3, v4, v5, on_failure)\
  GTEST_ASSERT_(::testing::AssertPred5Helper(#pred, \
                                             #v1, \
                                             #v2, \
                                             #v3, \
                                             #v4, \
                                             #v5, \
                                             pred, \
                                             v1, \
                                             v2, \
                                             v3, \
                                             v4, \
                                             v5), on_failure)

// 5-ary predicate assertion macros.
#define EXPECT_PRED_FORMAT5(pred_format, v1, v2, v3, v4, v5) \
  GTEST_PRED_FORMAT5_(pred_format, v1, v2, v3, v4, v5, GTEST_NONFATAL_FAILURE_)
#define EXPECT_PRED5(pred, v1, v2, v3, v4, v5) \
  GTEST_PRED5_(pred, v1, v2, v3, v4, v5, GTEST_NONFATAL_FAILURE_)
#define ASSERT_PRED_FORMAT5(pred_format, v1, v2, v3, v4, v5) \
  GTEST_PRED_FORMAT5_(pred_format, v1, v2, v3, v4, v5, GTEST_FATAL_FAILURE_)
#define ASSERT_PRED5(pred, v1, v2, v3, v4, v5) \
  GTEST_PRED5_(pred, v1, v2, v3, v4, v5, GTEST_FATAL_FAILURE_)



}  // namespace testing

#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.

```
// Copyright 2006, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

//
// Google C++ Testing and Mocking Framework definitions useful in production code.
// GOOGLETEST_CM0003 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_GTEST_PROD_H_
#define GTEST_INCLUDE_GTEST_GTEST_PROD_H_

// When you need to test the private or protected members of a class,
// use the FRIEND_TEST macro to declare your tests as friends of the
// class.  For example:
//
// class MyClass {
//  private:
//   void PrivateMethod();
//   FRIEND_TEST(MyClassTest, PrivateMethodWorks);
// };
//
// class MyClassTest : public testing::Test {
//   // ...
// };
//
// TEST_F(MyClassTest, PrivateMethodWorks) {
//   // Can call MyClass::PrivateMethod() here.
// }
//
// Note: The test class must be in the same namespace as the class being tested.
// For example, putting MyClassTest in an anonymous namespace will not work.

#define FRIEND_TEST(test_case_name, test_name)\
friend class test_case_name##_##test_name##_Test

#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.

```
// Copyright 2015, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// Injection point for custom user configurations. See README for details
//
// ** Custom implementation starts here **

#ifndef GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_PORT_H_
#define GTEST_INCLUDE_GTEST_INTERNAL_CUSTOM_GTEST_PORT_H_

#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_STRING_USES_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::GetArgvs ()
- 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)
```

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_RE

```
#define GTEST_HAS_POSIX_RE (!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 GTEST_USES_POSIX_RE

```
#define GTEST_USES_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.

```
// Copyright 2005, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// Low-level types and utilities for porting Google Test to various
// platforms.  All macros ending with _ and symbols defined in an
// internal namespace are subject to change without notice.  Code
// outside Google Test MUST NOT USE THEM DIRECTLY.  Macros that don't
// end with _ are part of Google Test's public API and can be used by
// code outside Google Test.
//
```

```
// This file is fundamental to Google Test.  All other Google Test source
// files are expected to #include this.  Therefore, it cannot #include
// any other Google Test header.

// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PORT_H_
#define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PORT_H_

// Environment-describing macros
// -----------------------------
//
// Google Test can be used in many different environments.  Macros in
// this section tell Google Test what kind of environment it is being
// used in, such that Google Test can provide environment-specific
// features and implementations.
//
// Google Test tries to automatically detect the properties of its
// environment, so users usually don't need to worry about these
// macros.  However, the automatic detection is not perfect.
// Sometimes it's necessary for a user to define some of the following
// macros in the build script to override Google Test's decisions.
//
// If the user doesn't define a macro in the list, Google Test will
// provide a default definition.  After this header is #included, all
// macros in this list will be defined to either 1 or 0.
//
// Notes to maintainers:
//   - Each macro here is a user-tweakable knob; do not grow the list
//     lightly.
//   - Use #if to key off these macros.  Don't use #ifdef or "#if
//     defined(...)", which will not work as these macros are ALWAYS
//     defined.
//
//   GTEST_HAS_CLONE          - Define it to 1/0 to indicate that clone(2)
//                              is/isn't available.
//   GTEST_HAS_EXCEPTIONS     - Define it to 1/0 to indicate that exceptions
//                              are enabled.
//   GTEST_HAS_GLOBAL_STRING  - Define it to 1/0 to indicate that ::string
//                              is/isn't available
//   GTEST_HAS_GLOBAL_WSTRING - Define it to 1/0 to indicate that ::wstring
//                              is/isn't available
//   GTEST_HAS_POSIX_RE       - Define it to 1/0 to indicate that POSIX regular
//                              expressions are/aren't available.
//   GTEST_HAS_PTHREAD        - Define it to 1/0 to indicate that <pthread.h>
//                              is/isn't available.
//   GTEST_HAS_RTTI           - Define it to 1/0 to indicate that RTTI is/isn't
//                              enabled.
//   GTEST_HAS_STD_WSTRING    - Define it to 1/0 to indicate that
//                              std::wstring does/doesn't work (Google Test can
//                              be used where std::wstring is unavailable).
//   GTEST_HAS_TR1_TUPLE      - Define it to 1/0 to indicate tr1::tuple
//                              is/isn't available.
//   GTEST_HAS_SEH            - Define it to 1/0 to indicate whether the
//                              compiler supports Microsoft's "Structured
//                              Exception Handling".
//   GTEST_HAS_STREAM_REDIRECTION
//                            - Define it to 1/0 to indicate whether the
//                              platform supports I/O stream redirection using
//                              dup() and dup2().
//   GTEST_USE_OWN_TR1_TUPLE  - Define it to 1/0 to indicate whether Google
//                              Test's own tr1 tuple implementation should be
//                              used.  Unused when the user sets
//                              GTEST_HAS_TR1_TUPLE to 0.
//   GTEST_LANG_CXX11         - Define it to 1/0 to indicate that Google Test
//                              is building in C++11/C++98 mode.
//   GTEST_LINKED_AS_SHARED_LIBRARY
//                            - Define to 1 when compiling tests that use
//                              Google Test as a shared library (known as
//                              DLL on Windows).
//   GTEST_CREATE_SHARED_LIBRARY
//                            - Define to 1 when compiling Google Test itself
//                              as a shared library.
//   GTEST_DEFAULT_DEATH_TEST_STYLE
//                            - The default value of --gtest_death_test_style.
//                              The legacy default has been "fast" in the open
//                              source version since 2008. The recommended value
//                              is "threadsafe", and can be set in
//                              custom/gtest-port.h.

// Platform-indicating macros
// --------------------------
//
// Macros indicating the platform on which Google Test is being used
// (a macro is defined to 1 if compiled on the given platform;
// otherwise UNDEFINED -- it's never defined to 0.).  Google Test
// defines these macros automatically.  Code outside Google Test MUST
// NOT define them.
//
//   GTEST_OS_AIX      - IBM AIX
//   GTEST_OS_CYGWIN   - Cygwin
//   GTEST_OS_FREEBSD  - FreeBSD
//   GTEST_OS_FUCHSIA  - Fuchsia
//   GTEST_OS_HPUX     - HP-UX
//   GTEST_OS_LINUX    - Linux
//     GTEST_OS_LINUX_ANDROID - Google Android
//   GTEST_OS_MAC      - Mac OS X
//     GTEST_OS_IOS    - iOS
//   GTEST_OS_NACL     - Google Native Client (NaCl)
//   GTEST_OS_NETBSD   - NetBSD
//   GTEST_OS_OPENBSD  - OpenBSD
//   GTEST_OS_QNX      - QNX
//   GTEST_OS_SOLARIS  - Sun Solaris
//   GTEST_OS_SYMBIAN  - Symbian
//   GTEST_OS_WINDOWS  - Windows (Desktop, MinGW, or Mobile)
//     GTEST_OS_WINDOWS_DESKTOP  - Windows Desktop
//     GTEST_OS_WINDOWS_MINGW    - MinGW
//     GTEST_OS_WINDOWS_MOBILE   - Windows Mobile
//     GTEST_OS_WINDOWS_PHONE    - Windows Phone
//     GTEST_OS_WINDOWS_RT       - Windows Store App/WinRT
//   GTEST_OS_ZOS      - z/OS
//
// Among the platforms, Cygwin, Linux, Max OS X, and Windows have the
// most stable support.  Since core members of the Google Test project
// don't have access to other platforms, support for them may be less
// stable.  If you notice any problems on your platform, please notify
// googletestframework@googlegroups.com (patches for fixing them are
// even more welcome!).
//
// It is possible that none of the GTEST_OS_* macros are defined.

// Feature-indicating macros
// -------------------------
//
// Macros indicating which Google Test features are available (a macro
// is defined to 1 if the corresponding feature is supported;
// otherwise UNDEFINED -- it's never defined to 0.).  Google Test
// defines these macros automatically.  Code outside Google Test MUST
// NOT define them.
//
// These macros are public so that portable tests can be written.
// Such tests typically surround code using a feature with an #if
// which controls that code.  For example:
//
// #if GTEST_HAS_DEATH_TEST
//   EXPECT_DEATH(DoSomethingDeadly());
// #endif
//
//   GTEST_HAS_COMBINE      - the Combine() function (for value-parameterized
//                            tests)
//   GTEST_HAS_DEATH_TEST   - death tests
//   GTEST_HAS_TYPED_TEST   - typed tests
//   GTEST_HAS_TYPED_TEST_P - type-parameterized tests
//   GTEST_IS_THREADSAFE    - Google Test is thread-safe.
//   GOOGLETEST_CM0007 DO NOT DELETE
//   GTEST_USES_POSIX_RE    - enhanced POSIX regex is used. Do not confuse with
//                            GTEST_HAS_POSIX_RE (see above) which users can
//                            define themselves.
//   GTEST_USES_SIMPLE_RE   - our own simple regex is used;
//                            the above RE\b(s) are mutually exclusive.
//   GTEST_CAN_COMPARE_NULL - accepts untyped NULL in EXPECT_EQ().

// Misc public macros
// ------------------
//
//   GTEST_FLAG(flag_name)  - references the variable corresponding to
//                            the given Google Test flag.

// Internal utilities
// ------------------
//
// The following macros and utilities are for Google Test's INTERNAL
// use only.  Code outside Google Test MUST NOT USE THEM DIRECTLY.
//
// Macros for basic C++ coding:
//   GTEST_AMBIGUOUS_ELSE_BLOCKER_ - for disabling a gcc warning.
//   GTEST_ATTRIBUTE_UNUSED_  - declares that a class' instances or a
//                              variable don't have to be used.
//   GTEST_DISALLOW_ASSIGN_   - disables operator=.
//   GTEST_DISALLOW_COPY_AND_ASSIGN_ - disables copy ctor and operator=.
//   GTEST_MUST_USE_RESULT_   - declares that a function's result must be used.
//   GTEST_INTENTIONAL_CONST_COND_PUSH_ - start code section where MSVC C4127 is
//                                        suppressed (constant conditional).
//   GTEST_INTENTIONAL_CONST_COND_POP_  - finish code section where MSVC C4127
//                                        is suppressed.
//
// C++11 feature wrappers:
//
//   testing::internal::forward - portability wrapper for std::forward.
//   testing::internal::move  - portability wrapper for std::move.
//
// Synchronization:
//   Mutex, MutexLock, ThreadLocal, GetThreadCount()
//                            - synchronization primitives.
//
// Template meta programming:
//   is_pointer     - as in TR1; needed on Symbian and IBM XL C/C++ only.
//   IteratorTraits - partial implementation of std::iterator_traits, which
//                    is not available in libCstd when compiled with Sun C++.
//
// Smart pointers:
//   scoped_ptr     - as in TR2.
//
// Regular expressions:
//   RE             - a simple regular expression class using the POSIX
//                    Extended Regular Expression syntax on UNIX-like platforms
//                    GOOGLETEST_CM0008 DO NOT DELETE
//                    or a reduced regular exception syntax on other
//                    platforms, including Windows.
// Logging:
//   GTEST_LOG_()   - logs messages at the specified severity level.
//   LogToStderr()  - directs all log messages to stderr.
//   FlushInfoLog() - flushes informational log messages.
//
// Stdout and stderr capturing:
//   CaptureStdout()     - starts capturing stdout.
//   GetCapturedStdout() - stops capturing stdout and returns the captured
//                         string.
//   CaptureStderr()     - starts capturing stderr.
//   GetCapturedStderr() - stops capturing stderr and returns the captured
//                         string.
//
// Integer types:
//   TypeWithSize   - maps an integer to a int type.
//   Int32, UInt32, Int64, UInt64, TimeInMillis
//                  - integers of known sizes.
//   BiggestInt     - the biggest signed integer type.
//
// Command-line utilities:
//   GTEST_DECLARE_*()  - declares a flag.
//   GTEST_DEFINE_*()   - defines a flag.
//   GetInjectableArgvs() - returns the command line as a vector of strings.
//
// Environment variable utilities:
//   GetEnv()             - gets the value of an environment variable.
//   BoolFromGTestEnv()   - parses a bool environment variable.
//   Int32FromGTestEnv()  - parses an Int32 environment variable.
//   StringFromGTestEnv() - parses a string environment variable.

#include <ctype.h>   // for isspace, etc
#include <stddef.h>  // for ptrdiff_t
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#ifndef _WIN32_WCE
# include <sys/types.h>
# include <sys/stat.h>
#endif  // !_WIN32_WCE

#if defined __APPLE__
# include <AvailabilityMacros.h>
# include <TargetConditionals.h>
#endif

// Brings in the definition of HAS_GLOBAL_STRING.  This must be done
// BEFORE we test HAS_GLOBAL_STRING.
#include <string>  // NOLINT
#include <algorithm>  // NOLINT
#include <iostream>  // NOLINT
#include <sstream>  // NOLINT
#include <utility>
#include <vector>  // NOLINT

#include "gtest/internal/gtest-port-arch.h"
#include "gtest/internal/custom/gtest-port.h"

#if !defined(GTEST_DEV_EMAIL_)
# 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/"
#endif  // !defined(GTEST_DEV_EMAIL_)

#if !defined(GTEST_INIT_GOOGLE_TEST_NAME_)
# define GTEST_INIT_GOOGLE_TEST_NAME_ "testing::InitGoogleTest"
#endif  // !defined(GTEST_INIT_GOOGLE_TEST_NAME_)

// Determines the version of gcc that is used to compile this.
#ifdef __GNUC__
// 40302 means version 4.3.2.
# define GTEST_GCC_VER_ \
    (__GNUC__*10000 + __GNUC_MINOR__*100 + __GNUC_PATCHLEVEL__)
#endif  // __GNUC__

// Macros for disabling Microsoft Visual C++ warnings.
//
//   GTEST_DISABLE_MSC_WARNINGS_PUSH_(4800 4385)
//   /* code that triggers warnings C4800 and C4385 */
//   GTEST_DISABLE_MSC_WARNINGS_POP_()
#if _MSC_VER >= 1400
# define GTEST_DISABLE_MSC_WARNINGS_PUSH_(warnings) \
    __pragma(warning(push))                        \
    __pragma(warning(disable: warnings))
# define GTEST_DISABLE_MSC_WARNINGS_POP_()          \
    __pragma(warning(pop))
#else
// Older versions of MSVC don't have __pragma.
# define GTEST_DISABLE_MSC_WARNINGS_PUSH_(warnings)
# define GTEST_DISABLE_MSC_WARNINGS_POP_()
#endif

// Clang on Windows does not understand MSVC's pragma warning.
// We need clang-specific way to disable function deprecation warning.
#ifdef __clang__
# define GTEST_DISABLE_MSC_DEPRECATED_PUSH_()                         \
    _Pragma("clang diagnostic push")                                  \
    _Pragma("clang diagnostic ignored \"-Wdeprecated-declarations\"") \
    _Pragma("clang diagnostic ignored \"-Wdeprecated-implementations\"")
#define GTEST_DISABLE_MSC_DEPRECATED_POP_() \
    _Pragma("clang diagnostic pop")
#else
# define GTEST_DISABLE_MSC_DEPRECATED_PUSH_() \
    GTEST_DISABLE_MSC_WARNINGS_PUSH_(4996)
# define GTEST_DISABLE_MSC_DEPRECATED_POP_() \
    GTEST_DISABLE_MSC_WARNINGS_POP_()
#endif

#ifndef GTEST_LANG_CXX11
// gcc and clang define __GXX_EXPERIMENTAL_CXX0X__ when
// -std={c,gnu}++{0x,11} is passed.  The C++11 standard specifies a
// value for __cplusplus, and recent versions of clang, gcc, and
// probably other compilers set that too in C++11 mode.
# if __GXX_EXPERIMENTAL_CXX0X__ || __cplusplus >= 201103L || _MSC_VER >= 1900
// Compiling in at least C++11 mode.
#  define GTEST_LANG_CXX11 1
# else
#  define GTEST_LANG_CXX11 0
# endif
#endif

// Distinct from C++11 language support, some environments don't provide
// proper C++11 library support. Notably, it's possible to build in
// C++11 mode when targeting Mac OS X 10.6, which has an old libstdc++
// with no C++11 support.
//
// libstdc++ has sufficient C++11 support as of GCC 4.6.0, __GLIBCXX__
// 20110325, but maintenance releases in the 4.4 and 4.5 series followed
// this date, so check for those versions by their date stamps.
// https://gcc.gnu.org/onlinedocs/libstdc++/manual/abi.html#abi.versioning
#if GTEST_LANG_CXX11 && \
    (!defined(__GLIBCXX__) || ( \
        __GLIBCXX__ >= 20110325ul &&  /* GCC >= 4.6.0 */ \
        /* Denylist of patch releases of older branches: */ \
        __GLIBCXX__ != 20110416ul &&  /* GCC 4.4.6 */ \
        __GLIBCXX__ != 20120313ul &&  /* GCC 4.4.7 */ \
        __GLIBCXX__ != 20110428ul &&  /* GCC 4.5.3 */ \
        __GLIBCXX__ != 20120702ul))   /* GCC 4.5.4 */
# define GTEST_STDLIB_CXX11 1
#endif

// Only use C++11 library features if the library provides them.
#if GTEST_STDLIB_CXX11
# define GTEST_HAS_STD_BEGIN_AND_END_ 1
# define GTEST_HAS_STD_FORWARD_LIST_ 1
# if !defined(_MSC_VER) || (_MSC_FULL_VER >= 190023824)
// works only with VS2015U2 and better
```

```
#   define GTEST_HAS_STD_FUNCTION_ 1
# endif
# define GTEST_HAS_STD_INITIALIZER_LIST_ 1
# define GTEST_HAS_STD_MOVE_ 1
# define GTEST_HAS_STD_UNIQUE_PTR_ 1
# define GTEST_HAS_STD_SHARED_PTR_ 1
# define GTEST_HAS_UNORDERED_MAP_ 1
# define GTEST_HAS_UNORDERED_SET_ 1
#endif

// C++11 specifies that <tuple> provides std::tuple.
// Some platforms still might not have it, however.
#if GTEST_LANG_CXX11
# define GTEST_HAS_STD_TUPLE_ 1
# if defined(__clang__)
// Inspired by
// https://clang.llvm.org/docs/LanguageExtensions.html#include-file-checking-macros
#  if defined(__has_include) && !__has_include(<tuple>)
#   undef GTEST_HAS_STD_TUPLE_
#  endif
# elif defined(_MSC_VER)
// Inspired by boost/config/stdlib/dinkumware.hpp
#  if defined(_CPPLIB_VER) && _CPPLIB_VER < 520
#   undef GTEST_HAS_STD_TUPLE_
#  endif
# elif defined(__GLIBCXX__)
// Inspired by boost/config/stdlib/libstdcpp3.hpp,
// http://gcc.gnu.org/gcc-4.2/changes.html and
//
https://web.archive.org/web/20140227044429/gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt01ch01.html#manual.intro.statu
#  if __GNUC__ < 4 || (__GNUC__ == 4 && __GNUC_MINOR__ < 2)
#   undef GTEST_HAS_STD_TUPLE_
#  endif
# endif
#endif

// Brings in definitions for functions used in the testing::internal::posix
// namespace (read, write, close, chdir, isatty, stat). We do not currently
// use them on Windows Mobile.
#if GTEST_OS_WINDOWS
# if !GTEST_OS_WINDOWS_MOBILE
#  include <direct.h>
#  include <io.h>
# endif
// In order to avoid having to include <windows.h>, use forward declaration
#if GTEST_OS_WINDOWS_MINGW && !defined(__MINGW64_VERSION_MAJOR)
// MinGW defined _CRITICAL_SECTION and _RTL_CRITICAL_SECTION as two
// separate (equivalent) structs, instead of using typedef
typedef struct _CRITICAL_SECTION GTEST_CRITICAL_SECTION;
#else
// Assume CRITICAL_SECTION is a typedef of _RTL_CRITICAL_SECTION.
// This assumption is verified by
// WindowsTypesTest.CRITICAL_SECTIONIs_RTL_CRITICAL_SECTION.
typedef struct _RTL_CRITICAL_SECTION GTEST_CRITICAL_SECTION;
#endif
#else
// This assumes that non-Windows OSes provide unistd.h. For OSes where this
// is not the case, we need to include headers that provide the functions
// mentioned above.
# include <unistd.h>
# include <strings.h>
#endif  // GTEST_OS_WINDOWS

#if GTEST_OS_LINUX_ANDROID
// Used to define __ANDROID_API__ matching the target NDK API level.
#  include <android/api-level.h>  // NOLINT
#endif

// Defines this to true iff Google Test can use POSIX regular expressions.
#ifndef GTEST_HAS_POSIX_RE
# if GTEST_OS_LINUX_ANDROID
// On Android, <regex.h> is only available starting with Gingerbread.
#  define GTEST_HAS_POSIX_RE (__ANDROID_API__ >= 9)
# else
#  define GTEST_HAS_POSIX_RE (!GTEST_OS_WINDOWS)
# endif
#endif

#if GTEST_USES_PCRE
// The appropriate headers have already been included.

#elif GTEST_HAS_POSIX_RE

// On some platforms, <regex.h> needs someone to define size_t, and
// won't compile otherwise.  We can #include it here as we already
// included <stdlib.h>, which is guaranteed to define size_t through
// <stddef.h>.
```

```
# include <regex.h>  // NOLINT

# define GTEST_USES_POSIX_RE 1

#elif GTEST_OS_WINDOWS

// <regex.h> is not available on Windows.  Use our own simple regex
// implementation instead.
# define GTEST_USES_SIMPLE_RE 1

#else

// <regex.h> may not be available on this platform.  Use our own
// simple regex implementation instead.
# define GTEST_USES_SIMPLE_RE 1

#endif  // GTEST_USES_PCRE

#ifndef GTEST_HAS_EXCEPTIONS
// The user didn't tell us whether exceptions are enabled, so we need
// to figure it out.
# if defined(_MSC_VER) && defined(_CPPUNWIND)
// MSVC defines _CPPUNWIND to 1 iff exceptions are enabled.
#  define GTEST_HAS_EXCEPTIONS 1
# elif defined(__BORLANDC__)
// C++Builder's implementation of the STL uses the _HAS_EXCEPTIONS
// macro to enable exceptions, so we'll do the same.
// Assumes that exceptions are enabled by default.
#  ifndef _HAS_EXCEPTIONS
#   define _HAS_EXCEPTIONS 1
#  endif  // _HAS_EXCEPTIONS
#  define GTEST_HAS_EXCEPTIONS _HAS_EXCEPTIONS
# elif defined(__clang__)
// clang defines __EXCEPTIONS iff exceptions are enabled before clang 220714,
// but iff cleanups are enabled after that. In Obj-C++ files, there can be
// cleanups for ObjC exceptions which also need cleanups, even if C++ exceptions
// are disabled. clang has __has_feature(cxx_exceptions) which checks for C++
// exceptions starting at clang r206352, but which checked for cleanups prior to
// that. To reliably check for C++ exception availability with clang, check for
// __EXCEPTIONS && __has_feature(cxx_exceptions).
#  define GTEST_HAS_EXCEPTIONS (__EXCEPTIONS && __has_feature(cxx_exceptions))
# elif defined(__GNUC__) && __EXCEPTIONS
// gcc defines __EXCEPTIONS to 1 iff exceptions are enabled.
#  define GTEST_HAS_EXCEPTIONS 1
# elif defined(__SUNPRO_CC)
// Sun Pro CC supports exceptions.  However, there is no compile-time way of
// detecting whether they are enabled or not.  Therefore, we assume that
// they are enabled unless the user tells us otherwise.
#  define GTEST_HAS_EXCEPTIONS 1
# elif defined(__IBMCPP__) && __EXCEPTIONS
// xlC defines __EXCEPTIONS to 1 iff exceptions are enabled.
#  define GTEST_HAS_EXCEPTIONS 1
# elif defined(__HP_aCC)
// Exception handling is in effect by default in HP aCC compiler. It has to
// be turned of by +noeh compiler option if desired.
#  define GTEST_HAS_EXCEPTIONS 1
# else
// For other compilers, we assume exceptions are disabled to be
// conservative.
#  define GTEST_HAS_EXCEPTIONS 0
# endif  // defined(_MSC_VER) || defined(__BORLANDC__)
#endif  // GTEST_HAS_EXCEPTIONS

#if !defined(GTEST_HAS_STD_STRING)
// Even though we don't use this macro any longer, we keep it in case
// some clients still depend on it.
# define GTEST_HAS_STD_STRING 1
#elif !GTEST_HAS_STD_STRING
// The user told us that ::std::string isn't available.
# error "::std::string isn't available."
#endif  // !defined(GTEST_HAS_STD_STRING)

#ifndef GTEST_HAS_GLOBAL_STRING
# define GTEST_HAS_GLOBAL_STRING 0
#endif  // GTEST_HAS_GLOBAL_STRING

#ifndef GTEST_HAS_STD_WSTRING
// The user didn't tell us whether ::std::wstring is available, so we need
// to figure it out.
// FIXME: uses autoconf to detect whether ::std::wstring
//   is available.

// Cygwin 1.7 and below doesn't support ::std::wstring.
// Solaris' libc++ doesn't support it either.  Android has
// no support for it at least as recent as Froyo (2.2).
# define GTEST_HAS_STD_WSTRING \
    (!(GTEST_OS_LINUX_ANDROID || GTEST_OS_CYGWIN || GTEST_OS_SOLARIS))
```

```

#endif  // GTEST_HAS_STD_WSTRING

#ifndef GTEST_HAS_GLOBAL_WSTRING
// The user didn't tell us whether ::wstring is available, so we need
// to figure it out.
# define GTEST_HAS_GLOBAL_WSTRING \
    (GTEST_HAS_STD_WSTRING && GTEST_HAS_GLOBAL_STRING)
#endif  // GTEST_HAS_GLOBAL_WSTRING

// Determines whether RTTI is available.
#ifndef GTEST_HAS_RTTI
// The user didn't tell us whether RTTI is enabled, so we need to
// figure it out.

# ifdef _MSC_VER

#  ifdef _CPPRTTI  // MSVC defines this macro iff RTTI is enabled.
#   define GTEST_HAS_RTTI 1
#  else
#   define GTEST_HAS_RTTI 0
#  endif

// Starting with version 4.3.2, gcc defines __GXX_RTTI iff RTTI is enabled.
# elif defined(__GNUC__) && (GTEST_GCC_VER_ >= 40302)

#  ifdef __GXX_RTTI
// When building against STLport with the Android NDK and with
// -frtti -fno-exceptions, the build fails at link time with undefined
// references to __cxa_bad_typeid. Note sure if STL or toolchain bug,
// so disable RTTI when detected.
#   if GTEST_OS_LINUX_ANDROID && defined(_STLPORT_MAJOR) && \
       !defined(__EXCEPTIONS)
#    define GTEST_HAS_RTTI 0
#   else
#    define GTEST_HAS_RTTI 1
#   endif  // GTEST_OS_LINUX_ANDROID && __STLPORT_MAJOR && !__EXCEPTIONS
#  else
#   define GTEST_HAS_RTTI 0
#  endif  // __GXX_RTTI

// Clang defines __GXX_RTTI starting with version 3.0, but its manual recommends
// using has_feature instead. has_feature(cxx_rtti) is supported since 2.7, the
// first version with C++ support.
# elif defined(__clang__)

#  define GTEST_HAS_RTTI __has_feature(cxx_rtti)

// Starting with version 9.0 IBM Visual Age defines __RTTI_ALL__ to 1 if
// both the typeid and dynamic_cast features are present.
# elif defined(__IBMCPP__) && (__IBMCPP__ >= 900)

#  ifdef __RTTI_ALL__
#   define GTEST_HAS_RTTI 1
#  else
#   define GTEST_HAS_RTTI 0
#  endif

# else

// For all other compilers, we assume RTTI is enabled.
#  define GTEST_HAS_RTTI 1

# endif  // _MSC_VER

#endif  // GTEST_HAS_RTTI

// It's this header's responsibility to #include <typeinfo> when RTTI
// is enabled.
#if GTEST_HAS_RTTI
# include <typeinfo>
#endif

// Determines whether Google Test can use the pthreads library.
#ifndef GTEST_HAS_PTHREAD
// The user didn't tell us explicitly, so we make reasonable assumptions about
// which platforms have pthreads support.
//
// To disable threading support in Google Test, add -DGTEST_HAS_PTHREAD=0
// to your compiler flags.
#define GTEST_HAS_PTHREAD                                                 \
  (GTEST_OS_LINUX || GTEST_OS_MAC || GTEST_OS_HPUX || GTEST_OS_QNX || \
   GTEST_OS_FREEBSD || GTEST_OS_NACL || GTEST_OS_NETBSD || GTEST_OS_FUCHSIA)
#endif  // GTEST_HAS_PTHREAD

#if GTEST_HAS_PTHREAD
// gtest-port.h guarantees to #include <pthread.h> when GTEST_HAS_PTHREAD is
// true.
# include <pthread.h>  // NOLINT

// For timespec and nanosleep, used below.
# include <time.h>  // NOLINT
#endif

// Determines if hash_map/hash_set are available.
// Only used for testing against those containers.
#if !defined(GTEST_HAS_HASH_MAP_)
# if defined(_MSC_VER) && (_MSC_VER < 1900)
#  define GTEST_HAS_HASH_MAP_ 1  // Indicates that hash_map is available.
#  define GTEST_HAS_HASH_SET_ 1  // Indicates that hash_set is available.
# endif  // _MSC_VER
#endif  // !defined(GTEST_HAS_HASH_MAP_)

// Determines whether Google Test can use tr1/tuple.  You can define
// this macro to 0 to prevent Google Test from using tuple (any
// feature depending on tuple with be disabled in this mode).
#ifndef GTEST_HAS_TR1_TUPLE
# if GTEST_OS_LINUX_ANDROID && defined(_STLPORT_MAJOR)
// STLport, provided with the Android NDK, has neither <tr1/tuple> or <tuple>.
#  define GTEST_HAS_TR1_TUPLE 0
# elif defined(_MSC_VER) && (_MSC_VER >= 1910)
// Prevent `warning C4996: 'std::tr1': warning STL4002:
// The non-Standard std::tr1 namespace and TR1-only machinery
// are deprecated and will be REMOVED.`
#  define GTEST_HAS_TR1_TUPLE 0
# elif GTEST_LANG_CXX11 && defined(_LIBCPP_VERSION)
// libc++ doesn't support TR1.
#  define GTEST_HAS_TR1_TUPLE 0
# else
// The user didn't tell us not to do it, so we assume it's OK.
#  define GTEST_HAS_TR1_TUPLE 1
# endif
#endif  // GTEST_HAS_TR1_TUPLE

// Determines whether Google Test's own tr1 tuple implementation
// should be used.
#ifndef GTEST_USE_OWN_TR1_TUPLE
// We use our own tuple implementation on Symbian.
# if GTEST_OS_SYMBIAN
#  define GTEST_USE_OWN_TR1_TUPLE 1
# else
// The user didn't tell us, so we need to figure it out.

// We use our own TR1 tuple if we aren't sure the user has an
// implementation of it already.  At this time, libstdc++ 4.0.0+ and
// MSVC 2010 are the only mainstream standard libraries that come
// with a TR1 tuple implementation.  NVIDIA's CUDA NVCC compiler
// pretends to be GCC by defining __GNUC__ and friends, but cannot
// compile GCC's tuple implementation.  MSVC 2008 (9.0) provides TR1
// tuple in a 323 MB Feature Pack download, which we cannot assume the
// user has.  QNX's QCC compiler is a modified GCC but it doesn't
// support TR1 tuple.  libc++ only provides std::tuple, in C++11 mode,
// and it can be used with some compilers that define __GNUC__.
# if (defined(__GNUC__) && !defined(__CUDACC__) && (GTEST_GCC_VER_ >= 40000) \
      && !GTEST_OS_QNX && !defined(_LIBCPP_VERSION)) \
      || (_MSC_VER >= 1600 && _MSC_VER < 1900)
#  define GTEST_ENV_HAS_TR1_TUPLE_ 1
# endif

// C++11 specifies that <tuple> provides std::tuple. Use that if gtest is used
// in C++11 mode and libstdc++ isn't very old (binaries targeting OS X 10.6
// can build with clang but need to use gcc4.2's libstdc++).
# if GTEST_LANG_CXX11 && (!defined(__GLIBCXX__) || __GLIBCXX__ > 20110325)
#  define GTEST_ENV_HAS_STD_TUPLE_ 1
# endif

# if GTEST_ENV_HAS_TR1_TUPLE_ || GTEST_ENV_HAS_STD_TUPLE_
#  define GTEST_USE_OWN_TR1_TUPLE 0
# else
#  define GTEST_USE_OWN_TR1_TUPLE 1
# endif
# endif  // GTEST_OS_SYMBIAN
#endif  // GTEST_USE_OWN_TR1_TUPLE

// To avoid conditional compilation we make it gtest-port.h's responsibility
// to #include the header implementing tuple.
#if GTEST_HAS_STD_TUPLE_
# include <tuple>  // IWYU pragma: export
# define GTEST_TUPLE_NAMESPACE_ ::std
#endif  // GTEST_HAS_STD_TUPLE_

// We include tr1::tuple even if std::tuple is available to define printers for
// them.
#if GTEST_HAS_TR1_TUPLE
```

```
# ifndef GTEST_TUPLE_NAMESPACE_
#  define GTEST_TUPLE_NAMESPACE_ ::std::tr1
# endif  // GTEST_TUPLE_NAMESPACE_

# if GTEST_USE_OWN_TR1_TUPLE
#  include "gtest/internal/gtest-tuple.h"  // IWYU pragma: export  // NOLINT
# elif GTEST_OS_SYMBIAN

// On Symbian, BOOST_HAS_TR1_TUPLE causes Boost's TR1 tuple library to
// use STLport's tuple implementation, which unfortunately doesn't
// work as the copy of STLport distributed with Symbian is incomplete.
// By making sure BOOST_HAS_TR1_TUPLE is undefined, we force Boost to
// use its own tuple implementation.
#  ifdef BOOST_HAS_TR1_TUPLE
#   undef BOOST_HAS_TR1_TUPLE
#  endif  // BOOST_HAS_TR1_TUPLE

// This prevents <boost/tr1/detail/config.hpp>, which defines
// BOOST_HAS_TR1_TUPLE, from being #included by Boost's <tuple>.
#  define BOOST_TR1_DETAIL_CONFIG_HPP_INCLUDED
#  include <tuple>  // IWYU pragma: export  // NOLINT

# elif defined(__GNUC__) && (GTEST_GCC_VER_ >= 40000)
// GCC 4.0+ implements tr1/tuple in the <tr1/tuple> header.  This does
// not conform to the TR1 spec, which requires the header to be <tuple>.

#  if !GTEST_HAS_RTTI && GTEST_GCC_VER_ < 40302
// Until version 4.3.2, gcc has a bug that causes <tr1/functional>,
// which is #included by <tr1/tuple>, to not compile when RTTI is
// disabled.  _TR1_FUNCTIONAL is the header guard for
// <tr1/functional>.  Hence the following #define is used to prevent
// <tr1/functional> from being included.
#   define _TR1_FUNCTIONAL 1
#   include <tr1/tuple>
#   undef _TR1_FUNCTIONAL  // Allows the user to #include
                        // <tr1/functional> if they choose to.
#  else
#   include <tr1/tuple>  // NOLINT
#  endif  // !GTEST_HAS_RTTI && GTEST_GCC_VER_ < 40302

// VS 2010 now has tr1 support.
# elif _MSC_VER >= 1600
#  include <tuple>  // IWYU pragma: export  // NOLINT

# else  // GTEST_USE_OWN_TR1_TUPLE
#  include <tr1/tuple>  // IWYU pragma: export  // NOLINT
# endif  // GTEST_USE_OWN_TR1_TUPLE

#endif  // GTEST_HAS_TR1_TUPLE

// Determines whether clone(2) is supported.
// Usually it will only be available on Linux, excluding
// Linux on the Itanium architecture.
// Also see http://linux.die.net/man/2/clone.
#ifndef GTEST_HAS_CLONE
// The user didn't tell us, so we need to figure it out.

# if GTEST_OS_LINUX && !defined(__ia64__)
#  if GTEST_OS_LINUX_ANDROID
// On Android, clone() became available at different API levels for each 32-bit
// architecture.
#    if defined(__LP64__) || \
        (defined(__arm__) && __ANDROID_API__ >= 9) || \
        (defined(__mips__) && __ANDROID_API__ >= 12) || \
        (defined(__i386__) && __ANDROID_API__ >= 17)
#     define GTEST_HAS_CLONE 1
#    else
#     define GTEST_HAS_CLONE 0
#    endif
#  else
#   define GTEST_HAS_CLONE 1
#  endif
# else
#  define GTEST_HAS_CLONE 0
# endif  // GTEST_OS_LINUX && !defined(__ia64__)

#endif  // GTEST_HAS_CLONE

// Determines whether to support stream redirection. This is used to test
// output correctness and to implement death tests.
#ifndef GTEST_HAS_STREAM_REDIRECTION
// By default, we assume that stream redirection is supported on all
// platforms except known mobile ones.
# if GTEST_OS_WINDOWS_MOBILE || GTEST_OS_SYMBIAN || \
    GTEST_OS_WINDOWS_PHONE || GTEST_OS_WINDOWS_RT
#  define GTEST_HAS_STREAM_REDIRECTION 0
# else
#  define GTEST_HAS_STREAM_REDIRECTION 1
# endif  // !GTEST_OS_WINDOWS_MOBILE && !GTEST_OS_SYMBIAN
#endif  // GTEST_HAS_STREAM_REDIRECTION

// Determines whether to support death tests.
// Google Test does not support death tests for VC 7.1 and earlier as
// abort() in a VC 7.1 application compiled as GUI in debug config
// pops up a dialog window that cannot be suppressed programmatically.
#if (GTEST_OS_LINUX || GTEST_OS_CYGWIN || GTEST_OS_SOLARIS ||   \
     (GTEST_OS_MAC && !GTEST_OS_IOS) ||                         \
     (GTEST_OS_WINDOWS_DESKTOP && _MSC_VER >= 1400) ||          \
     GTEST_OS_WINDOWS_MINGW || GTEST_OS_AIX || GTEST_OS_HPUX || \
     GTEST_OS_OPENBSD || GTEST_OS_QNX || GTEST_OS_FREEBSD || \
     GTEST_OS_NETBSD || GTEST_OS_FUCHSIA)
# define GTEST_HAS_DEATH_TEST 1
#endif

// Determines whether to support type-driven tests.

// Typed tests need <typeinfo> and variadic macros, which GCC, VC++ 8.0,
// Sun Pro CC, IBM Visual Age, and HP aCC support.
#if defined(__GNUC__) || (_MSC_VER >= 1400) || defined(__SUNPRO_CC) || \
    defined(__IBMCPP__) || defined(__HP_aCC)
# define GTEST_HAS_TYPED_TEST 1
# define GTEST_HAS_TYPED_TEST_P 1
#endif

// Determines whether to support Combine(). This only makes sense when
// value-parameterized tests are enabled.  The implementation doesn’t
// work on Sun Studio since it doesn’t understand templated conversion
// operators.
#if (GTEST_HAS_TR1_TUPLE || GTEST_HAS_STD_TUPLE_) && !defined(__SUNPRO_CC)
# define GTEST_HAS_COMBINE 1
#endif

// Determines whether the system compiler uses UTF-16 for encoding wide strings.
#define GTEST_WIDE_STRING_USES_UTF16_ \
    (GTEST_OS_WINDOWS || GTEST_OS_CYGWIN || GTEST_OS_SYMBIAN || GTEST_OS_AIX)

// Determines whether test results can be streamed to a socket.
#if GTEST_OS_LINUX
# define GTEST_CAN_STREAM_RESULTS_ 1
#endif

// Defines some utility macros.

// The GNU compiler emits a warning if nested "if" statements are followed by
// an "else" statement and braces are not used to explicitly disambiguate the
// "else" binding.  This leads to problems with code like:
//
//   if (gate)
//     ASSERT_*(condition) « "Some message";
//
// The "switch (0) case 0:" idiom is used to suppress this.
#ifdef __INTEL_COMPILER
# define GTEST_AMBIGUOUS_ELSE_BLOCKER_
#else
# define GTEST_AMBIGUOUS_ELSE_BLOCKER_ switch (0) case 0: default:  // NOLINT
#endif

// Use this annotation at the end of a struct/class definition to
// prevent the compiler from optimizing away instances that are never
// used.  This is useful when all interesting logic happens inside the
// c’tor and / or d’tor.  Example:
//
//   struct Foo {
//     Foo() { ... }
//   } GTEST_ATTRIBUTE_UNUSED_;
//
// Also use it after a variable or parameter declaration to tell the
// compiler the variable/parameter does not have to be used.
#if defined(__GNUC__) && !defined(COMPILER_ICC)
# define GTEST_ATTRIBUTE_UNUSED_ __attribute__ ((unused))
#elif defined(__clang__)
# if __has_attribute(unused)
#  define GTEST_ATTRIBUTE_UNUSED_ __attribute__ ((unused))
# endif
#endif
#ifndef GTEST_ATTRIBUTE_UNUSED_
# define GTEST_ATTRIBUTE_UNUSED_
#endif

#if GTEST_LANG_CXX11
# define GTEST_CXX11_EQUALS_DELETE_ = delete
#else  // GTEST_LANG_CXX11
# define GTEST_CXX11_EQUALS_DELETE_
#endif  // GTEST_LANG_CXX11
```

```

// Use this annotation before a function that takes a printf format string.
#if (defined(__GNUC__) || defined(__clang__)) && !defined(COMPILER_ICC)
# if defined(__MINGW_PRINTF_FORMAT)
// MinGW has two different printf implementations. Ensure the format macro
// matches the selected implementation. See
// https://sourceforge.net/p/mingw-w64/wiki2/gnu%20printf/.
#  define GTEST_ATTRIBUTE_PRINTF_(string_index, first_to_check) \
       __attribute__((__format__(__MINGW_PRINTF_FORMAT, string_index, \
                                 first_to_check)))
# else
#  define GTEST_ATTRIBUTE_PRINTF_(string_index, first_to_check) \
       __attribute__((__format__(__printf__, string_index, first_to_check)))
# endif
#else
# define GTEST_ATTRIBUTE_PRINTF_(string_index, first_to_check)
#endif


// A macro to disallow operator=
// This should be used in the private: declarations for a class.
#define GTEST_DISALLOW_ASSIGN_(type) \
  void operator=(type const &) GTEST_CXX11_EQUALS_DELETE_

// A macro to disallow copy constructor and operator=
// This should be used in the private: declarations for a class.
#define GTEST_DISALLOW_COPY_AND_ASSIGN_(type) \
  type(type const &) GTEST_CXX11_EQUALS_DELETE_; \
  GTEST_DISALLOW_ASSIGN_(type)

// Tell the compiler to warn about unused return values for functions declared
// with this macro.  The macro should be used on function declarations
// following the argument list:
//
//   Sprocket* AllocateSprocket() GTEST_MUST_USE_RESULT_;
#if defined(__GNUC__) && (GTEST_GCC_VER_ >= 30400) && !defined(COMPILER_ICC)
# define GTEST_MUST_USE_RESULT_ __attribute__ ((warn_unused_result))
#else
# define GTEST_MUST_USE_RESULT_
#endif  // __GNUC__ && (GTEST_GCC_VER_ >= 30400) && !COMPILER_ICC

// MS C++ compiler emits warning when a conditional expression is compile time
// constant. In some contexts this warning is false positive and needs to be
// suppressed. Use the following two macros in such cases:
//
// GTEST_INTENTIONAL_CONST_COND_PUSH_()
// while (true) {
// GTEST_INTENTIONAL_CONST_COND_POP_()
// }
# define GTEST_INTENTIONAL_CONST_COND_PUSH_() \
    GTEST_DISABLE_MSC_WARNINGS_PUSH_(4127)
# define GTEST_INTENTIONAL_CONST_COND_POP_() \
    GTEST_DISABLE_MSC_WARNINGS_POP_()

// Determine whether the compiler supports Microsoft's Structured Exception
// Handling.  This is supported by several Windows compilers but generally
// does not exist on any other system.
#ifndef GTEST_HAS_SEH
// The user didn't tell us, so we need to figure it out.

# if defined(_MSC_VER) || defined(__BORLANDC__)
// These two compilers are known to support SEH.
#  define GTEST_HAS_SEH 1
# else
// Assume no SEH.
#  define GTEST_HAS_SEH 0
# endif

#define GTEST_IS_THREADSAFE \
    (GTEST_HAS_MUTEX_AND_THREAD_LOCAL_ \
     || (GTEST_OS_WINDOWS && !GTEST_OS_WINDOWS_PHONE && !GTEST_OS_WINDOWS_RT) \
     || GTEST_HAS_PTHREAD)

#endif  // GTEST_HAS_SEH

// GTEST_API_ qualifies all symbols that must be exported. The definitions below
// are guarded by #ifndef to give embedders a chance to define GTEST_API_ in
// gtest/internal/custom/gtest-port.h
#ifndef GTEST_API_

#ifdef _MSC_VER
# if GTEST_LINKED_AS_SHARED_LIBRARY
#  define GTEST_API_ __declspec(dllimport)
# elif GTEST_CREATE_SHARED_LIBRARY
#  define GTEST_API_ __declspec(dllexport)
# endif
#elif __GNUC__ >= 4 || defined(__clang__)
```

```
# define GTEST_API_ __attribute__((visibility ("default")))
#endif  // _MSC_VER

#endif  // GTEST_API_

#ifndef GTEST_API_
# define GTEST_API_
#endif  // GTEST_API_

#ifndef GTEST_DEFAULT_DEATH_TEST_STYLE
# define GTEST_DEFAULT_DEATH_TEST_STYLE  "fast"
#endif  // GTEST_DEFAULT_DEATH_TEST_STYLE

#ifdef __GNUC__
// Ask the compiler to never inline a given function.
# define GTEST_NO_INLINE_ __attribute__((noinline))
#else
# define GTEST_NO_INLINE_
#endif

// _LIBCPP_VERSION is defined by the libc++ library from the LLVM project.
#if !defined(GTEST_HAS_CXXABI_H_)
# if defined(__GLIBCXX__) || (defined(_LIBCPP_VERSION) && !defined(_MSC_VER))
#  define GTEST_HAS_CXXABI_H_ 1
# else
#  define GTEST_HAS_CXXABI_H_ 0
# endif
#endif

// A function level attribute to disable checking for use of uninitialized
// memory when built with MemorySanitizer.
#if defined(__clang__)
# if __has_feature(memory_sanitizer)
#  define GTEST_ATTRIBUTE_NO_SANITIZE_MEMORY_ \
       __attribute__((no_sanitize_memory))
# else
#  define GTEST_ATTRIBUTE_NO_SANITIZE_MEMORY_
# endif  // __has_feature(memory_sanitizer)
#else
# define GTEST_ATTRIBUTE_NO_SANITIZE_MEMORY_
#endif  // __clang__

// A function level attribute to disable AddressSanitizer instrumentation.
#if defined(__clang__)
# if __has_feature(address_sanitizer)
#  define GTEST_ATTRIBUTE_NO_SANITIZE_ADDRESS_ \
       __attribute__((no_sanitize_address))
# else
#  define GTEST_ATTRIBUTE_NO_SANITIZE_ADDRESS_
# endif  // __has_feature(address_sanitizer)
#else
# define GTEST_ATTRIBUTE_NO_SANITIZE_ADDRESS_
#endif  // __clang__

// A function level attribute to disable ThreadSanitizer instrumentation.
#if defined(__clang__)
# if __has_feature(thread_sanitizer)
#  define GTEST_ATTRIBUTE_NO_SANITIZE_THREAD_ \
       __attribute__((no_sanitize_thread))
# else
#  define GTEST_ATTRIBUTE_NO_SANITIZE_THREAD_
# endif  // __has_feature(thread_sanitizer)
#else
# define GTEST_ATTRIBUTE_NO_SANITIZE_THREAD_
#endif  // __clang__

namespace testing {

class Message;

#if defined(GTEST_TUPLE_NAMESPACE_)
// Import tuple and friends into the ::testing namespace.
// It is part of our interface, having them in ::testing allows us to change
// their types as needed.
using GTEST_TUPLE_NAMESPACE_::get;
using GTEST_TUPLE_NAMESPACE_::make_tuple;
using GTEST_TUPLE_NAMESPACE_::tuple;
using GTEST_TUPLE_NAMESPACE_::tuple_size;
using GTEST_TUPLE_NAMESPACE_::tuple_element;
#endif  // defined(GTEST_TUPLE_NAMESPACE_)

namespace internal {

// A secret type that Google Test users don't know about.  It has no
// definition on purpose.  Therefore it's impossible to create a
// Secret object, which is what we want.
class Secret;
```

```

// The GTEST_COMPILE_ASSERT_ macro can be used to verify that a compile time
// expression is true. For example, you could use it to verify the
// size of a static array:
//
//   GTEST_COMPILE_ASSERT_(GTEST_ARRAY_SIZE_(names) == NUM_NAMES,
//                         names_incorrect_size);
//
// or to make sure a struct is smaller than a certain size:
//
//   GTEST_COMPILE_ASSERT_(sizeof(foo) < 128, foo_too_large);
//
// The second argument to the macro is the name of the variable. If
// the expression is false, most compilers will issue a warning/error
// containing the name of the variable.

#if GTEST_LANG_CXX11
# define GTEST_COMPILE_ASSERT_(expr, msg) static_assert(expr, #msg)
#else  // !GTEST_LANG_CXX11
template <bool>
  struct CompileAssert {
};

# define GTEST_COMPILE_ASSERT_(expr, msg) \
  typedef ::testing::internal::CompileAssert<(static_cast<bool>(expr))> \
      msg[static_cast<bool>(expr) ? 1 : -1] GTEST_ATTRIBUTE_UNUSED_
#endif  // !GTEST_LANG_CXX11

// Implementation details of GTEST_COMPILE_ASSERT_:
//
// (In C++11, we simply use static_assert instead of the following)
//
// - GTEST_COMPILE_ASSERT_ works by defining an array type that has -1
//   elements (and thus is invalid) when the expression is false.
//
// - The simpler definition
//
//    #define GTEST_COMPILE_ASSERT_(expr, msg) typedef char msg[(expr) ? 1 : -1]
//
//   does not work, as gcc supports variable-length arrays whose sizes
//   are determined at run-time (this is gcc's extension and not part
//   of the C++ standard).  As a result, gcc fails to reject the
//   following code with the simple definition:
//
//     int foo;
//     GTEST_COMPILE_ASSERT_(foo, msg); // not supposed to compile as foo is
//                                      // not a compile-time constant.
//
// - By using the type CompileAssert<(bool(expr))>, we ensures that
//   expr is a compile-time constant.  (Template arguments must be
//   determined at compile-time.)
//
// - The outter parentheses in CompileAssert<(bool(expr))> are necessary
//   to work around a bug in gcc 3.4.4 and 4.0.1.  If we had written
//
//     CompileAssert<bool(expr)>
//
//   instead, these compilers will refuse to compile
//
//     GTEST_COMPILE_ASSERT_(5 > 0, some_message);
//
//   (They seem to think the ">" in "5 > 0" marks the end of the
//   template argument list.)
//
// - The array size is (bool(expr) ? 1 : -1), instead of simply
//
//     ((expr) ? 1 : -1).
//
//   This is to avoid running into a bug in MS VC 7.1, which
//   causes ((0.0) ? 1 : -1) to incorrectly evaluate to 1.

// StaticAssertTypeEqHelper is used by StaticAssertTypeEq defined in gtest.h.
//
// This template is declared, but intentionally undefined.
template <typename T1, typename T2>
struct StaticAssertTypeEqHelper;

template <typename T>
struct StaticAssertTypeEqHelper<T, T> {
  enum { value = true };
};

// Same as std::is_same<>.
template <typename T, typename U>
struct IsSame {
  enum { value = false };
};
```

```
template <typename T>
struct IsSame<T, T> {
  enum { value = true };
};

// Evaluates to the number of elements in 'array'.
#define GTEST_ARRAY_SIZE_(array) (sizeof(array) / sizeof(array[0]))

#if GTEST_HAS_GLOBAL_STRING
typedef ::string string;
#else
typedef ::std::string string;
#endif  // GTEST_HAS_GLOBAL_STRING

#if GTEST_HAS_GLOBAL_WSTRING
typedef ::wstring wstring;
#elif GTEST_HAS_STD_WSTRING
typedef ::std::wstring wstring;
#endif  // GTEST_HAS_GLOBAL_WSTRING

// A helper for suppressing warnings on constant condition.  It just
// returns 'condition'.
GTEST_API_ bool IsTrue(bool condition);

// Defines scoped_ptr.

// This implementation of scoped_ptr is PARTIAL - it only contains
// enough stuff to satisfy Google Test's need.
template <typename T>
class scoped_ptr {
 public:
  typedef T element_type;

  explicit scoped_ptr(T* p = NULL) : ptr_(p) {}
  ~scoped_ptr() { reset(); }

  T& operator*() const { return *ptr_; }
  T* operator->() const { return ptr_; }
  T* get() const { return ptr_; }

  T* release() {
    T* const ptr = ptr_;
    ptr_ = NULL;
    return ptr;
  }

  void reset(T* p = NULL) {
    if (p != ptr_) {
      if (IsTrue(sizeof(T) > 0)) {  // Makes sure T is a complete type.
        delete ptr_;
      }
      ptr_ = p;
    }
  }

  friend void swap(scoped_ptr& a, scoped_ptr& b) {
    using std::swap;
    swap(a.ptr_, b.ptr_);
  }

 private:
  T* ptr_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(scoped_ptr);
};

// Defines RE.

#if GTEST_USES_PCRE
// if used, PCRE is injected by custom/gtest-port.h
#elif GTEST_USES_POSIX_RE || GTEST_USES_SIMPLE_RE

// A simple C++ wrapper for <regex.h>.  It uses the POSIX Extended
// Regular Expression syntax.
class GTEST_API_ RE {
 public:
  // A copy constructor is required by the Standard to initialize object
  // references from r-values.
  RE(const RE& other) { Init(other.pattern()); }

  // Constructs an RE from a string.
  RE(const ::std::string& regex) { Init(regex.c_str()); }  // NOLINT

# if GTEST_HAS_GLOBAL_STRING

  RE(const ::string& regex) { Init(regex.c_str()); }  // NOLINT

# endif  // GTEST_HAS_GLOBAL_STRING

  RE(const char* regex) { Init(regex); }  // NOLINT
  ~RE();

  // Returns the string representation of the regex.
  const char* pattern() const { return pattern_; }

  // FullMatch(str, re) returns true iff regular expression re matches
  // the entire str.
  // PartialMatch(str, re) returns true iff regular expression re
  // matches a substring of str (including str itself).
  //
  // FIXME: make FullMatch() and PartialMatch() work
  // when str contains NUL characters.
  static bool FullMatch(const ::std::string& str, const RE& re) {
    return FullMatch(str.c_str(), re);
  }
  static bool PartialMatch(const ::std::string& str, const RE& re) {
    return PartialMatch(str.c_str(), re);
  }

# if GTEST_HAS_GLOBAL_STRING

  static bool FullMatch(const ::string& str, const RE& re) {
    return FullMatch(str.c_str(), re);
  }
  static bool PartialMatch(const ::string& str, const RE& re) {
    return PartialMatch(str.c_str(), re);
  }

# endif  // GTEST_HAS_GLOBAL_STRING

  static bool FullMatch(const char* str, const RE& re);
  static bool PartialMatch(const char* str, const RE& re);

 private:
  void Init(const char* regex);

  // We use a const char* instead of an std::string, as Google Test used to be
  // used where std::string is not available.  FIXME: change to
  // std::string.
  const char* pattern_;
  bool is_valid_;

# if GTEST_USES_POSIX_RE

  regex_t full_regex_;     // For FullMatch().
  regex_t partial_regex_;  // For PartialMatch().

# else  // GTEST_USES_SIMPLE_RE

  const char* full_pattern_;  // For FullMatch();

# endif

  GTEST_DISALLOW_ASSIGN_(RE);
};

#endif  // GTEST_USES_PCRE

// Formats a source file path and a line number as they would appear
// in an error message from the compiler used to compile this code.
GTEST_API_ ::std::string FormatFileLocation(const char* file, int line);

// Formats a file location for compiler-independent XML output.
// Although this function is not platform dependent, we put it next to
// FormatFileLocation in order to contrast the two functions.
GTEST_API_ ::std::string FormatCompilerIndependentFileLocation(const char* file,
                                                               int line);

// Defines logging utilities:
//   GTEST_LOG_(severity) - logs messages at the specified severity level. The
//                          message itself is streamed into the macro.
//   LogToStderr()  - directs all log messages to stderr.
//   FlushInfoLog() - flushes informational log messages.

enum GTestLogSeverity {
  GTEST_INFO,
  GTEST_WARNING,
  GTEST_ERROR,
  GTEST_FATAL
};

// Formats log entry severity, provides a stream object for streaming the
// log message, and terminates the message with a newline when going out of
// scope.
```

```
class GTEST_API_ GTestLog {
 public:
  GTestLog(GTestLogSeverity severity, const char* file, int line);

  // Flushes the buffers and, if severity is GTEST_FATAL, aborts the program.
  ~GTestLog();

  ::std::ostream& GetStream() { return ::std::cerr; }

 private:
  const GTestLogSeverity severity_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(GTestLog);
};

#if !defined(GTEST_LOG_)

# define GTEST_LOG_(severity) \
    ::testing::internal::GTestLog(::testing::internal::GTEST_##severity, \
                                  __FILE__, __LINE__).GetStream()

inline void LogToStderr() {}
inline void FlushInfoLog() { fflush(NULL); }

#endif  // !defined(GTEST_LOG_)

#if !defined(GTEST_CHECK_)
// INTERNAL IMPLEMENTATION - DO NOT USE.
//
// GTEST_CHECK_ is an all-mode assert. It aborts the program if the condition
// is not satisfied.
//  Synopsys:
//    GTEST_CHECK_(boolean_condition);
//     or
//    GTEST_CHECK_(boolean_condition) « "Additional message";
//
//    This checks the condition and if the condition is not satisfied
//    it prints message about the condition violation, including the
//    condition itself, plus additional message streamed into it, if any,
//    and then it aborts the program. It aborts the program irrespective of
//    whether it is built in the debug mode or not.
# define GTEST_CHECK_(condition) \
    GTEST_AMBIGUOUS_ELSE_BLOCKER_ \
    if (::testing::internal::IsTrue(condition)) \
      ; \
    else \
      GTEST_LOG_(FATAL) « "Condition " #condition " failed. "
#endif  // !defined(GTEST_CHECK_)

// An all-mode assert to verify that the given POSIX-style function
// call returns 0 (indicating success).  Known limitation: this
// doesn't expand to a balanced 'if' statement, so enclose the macro
// in {} if you need to use it as the only statement in an 'if'
// branch.
#define GTEST_CHECK_POSIX_SUCCESS_(posix_call) \
  if (const int gtest_error = (posix_call)) \
    GTEST_LOG_(FATAL) « #posix_call « "failed with error " \
                      « gtest_error

// Adds reference to a type if it is not a reference type,
// otherwise leaves it unchanged.  This is the same as
// tr1::add_reference, which is not widely available yet.
template <typename T>
struct AddReference { typedef T& type; };  // NOLINT
template <typename T>
struct AddReference<T&> { typedef T& type; };  // NOLINT

// A handy wrapper around AddReference that works when the argument T
// depends on template parameters.
#define GTEST_ADD_REFERENCE_(T) \
    typename ::testing::internal::AddReference<T>::type

// Transforms "T" into "const T&" according to standard reference collapsing
// rules (this is only needed as a backport for C++98 compilers that do not
// support reference collapsing). Specifically, it transforms:
//
//   char         ==> const char&
//   const char   ==> const char&
//   char&        ==> char&
//   const char&  ==> const char&
//
// Note that the non-const reference will not have "const" added. This is
// standard, and necessary so that "T" can always bind to "const T&".
template <typename T>
struct ConstRef { typedef const T& type; };
template <typename T>
struct ConstRef<T&> { typedef T& type; };
```

```

// The argument T must depend on some template parameters.
#define GTEST_REFERENCE_TO_CONST_(T) \
  typename ::testing::internal::ConstRef<T>::type

#if GTEST_HAS_STD_MOVE_
using std::forward;
using std::move;

template <typename T>
struct RvalueRef {
  typedef T&& type;
};
#else  // GTEST_HAS_STD_MOVE_
template <typename T>
const T& move(const T& t) {
  return t;
}
template <typename T>
GTEST_ADD_REFERENCE_(T) forward(GTEST_ADD_REFERENCE_(T) t) { return t; }

template <typename T>
struct RvalueRef {
  typedef const T& type;
};
#endif  // GTEST_HAS_STD_MOVE_

// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
//
// Use ImplicitCast_ as a safe version of static_cast for upcasting in
// the type hierarchy (e.g. casting a Foo* to a SuperclassOfFoo* or a
// const Foo*).  When you use ImplicitCast_, the compiler checks that
// the cast is safe.  Such explicit ImplicitCast_s are necessary in
// surprisingly many situations where C++ demands an exact type match
// instead of an argument type convertable to a target type.
//
// The syntax for using ImplicitCast_ is the same as for static_cast:
//
//   ImplicitCast_<ToType>(expr)
//
// ImplicitCast_ would have been part of the C++ standard library,
// but the proposal was submitted too late.  It will probably make
// its way into the language in the future.
//
// This relatively ugly name is intentional. It prevents clashes with
// similar functions users may have (e.g., implicit_cast). The internal
// namespace alone is not enough because the function can be found by ADL.
template<typename To>
inline To ImplicitCast_(To x) { return x; }

// When you upcast (that is, cast a pointer from type Foo to type
// SuperclassOfFoo), it's fine to use ImplicitCast_<>, since upcasts
// always succeed.  When you downcast (that is, cast a pointer from
// type Foo to type SubclassOfFoo), static_cast<> isn't safe, because
// how do you know the pointer is really of type SubclassOfFoo?  It
// could be a bare Foo, or of type DifferentSubclassOfFoo.  Thus,
// when you downcast, you should use this macro.  In debug mode, we
// use dynamic_cast<> to double-check the downcast is legal (we die
// if it's not).  In normal mode, we do the efficient static_cast<>
// instead.  Thus, it's important to test in debug mode to make sure
// the cast is legal!
//    This is the only place in the code we should use dynamic_cast<>.
// In particular, you SHOULDN'T be using dynamic_cast<> in order to
// do RTTI (eg code like this:
//    if (dynamic_cast<Subclass1>(foo)) HandleASubclass1Object(foo);
//    if (dynamic_cast<Subclass2>(foo)) HandleASubclass2Object(foo);
// You should design the code some other way not to need this.
//
// This relatively ugly name is intentional. It prevents clashes with
// similar functions users may have (e.g., down_cast). The internal
// namespace alone is not enough because the function can be found by ADL.
template<typename To, typename From>  // use like this: DownCast_<T*>(foo);
inline To DownCast_(From* f) {  // so we only accept pointers
  // Ensures that To is a sub-type of From *.  This test is here only
  // for compile-time type checking, and has no overhead in an
  // optimized build at run-time, as it will be optimized away
  // completely.
  GTEST_INTENTIONAL_CONST_COND_PUSH_()
  if (false) {
  GTEST_INTENTIONAL_CONST_COND_POP_()
    const To to = NULL;
    ::testing::internal::ImplicitCast_<From*>(to);
  }

#if GTEST_HAS_RTTI
  // RTTI: debug mode only!
  GTEST_CHECK_(f == NULL || dynamic_cast<To>(f) != NULL);
```

```
#endif
  return static_cast<To>(f);
}

// Downcasts the pointer of type Base to Derived.
// Derived must be a subclass of Base. The parameter MUST
// point to a class of type Derived, not any subclass of it.
// When RTTI is available, the function performs a runtime
// check to enforce this.
template <class Derived, class Base>
Derived* CheckedDowncastToActualType(Base* base) {
#if GTEST_HAS_RTTI
  GTEST_CHECK_(typeid(*base) == typeid(Derived));
#endif

#if GTEST_HAS_DOWNCAST_
  return ::down_cast<Derived*>(base);
#elif GTEST_HAS_RTTI
  return dynamic_cast<Derived*>(base);  // NOLINT
#else
  return static_cast<Derived*>(base);  // Poor man's downcast.
#endif
}

#if GTEST_HAS_STREAM_REDIRECTION

// Defines the stderr capturer:
//   CaptureStdout     - starts capturing stdout.
//   GetCapturedStdout - stops capturing stdout and returns the captured string.
//   CaptureStderr     - starts capturing stderr.
//   GetCapturedStderr - stops capturing stderr and returns the captured string.
//
GTEST_API_ void CaptureStdout();
GTEST_API_ std::string GetCapturedStdout();
GTEST_API_ void CaptureStderr();
GTEST_API_ std::string GetCapturedStderr();

#endif  // GTEST_HAS_STREAM_REDIRECTION
// Returns the size (in bytes) of a file.
GTEST_API_ size_t GetFileSize(FILE* file);

// Reads the entire content of a file as a string.
GTEST_API_ std::string ReadEntireFile(FILE* file);

// All command line arguments.
GTEST_API_ std::vector<std::string> GetArgvs();

#if GTEST_HAS_DEATH_TEST

std::vector<std::string> GetInjectableArgvs();
// Deprecated: pass the args vector by value instead.
void SetInjectableArgvs(const std::vector<std::string>* new_argvs);
void SetInjectableArgvs(const std::vector<std::string>& new_argvs);
#if GTEST_HAS_GLOBAL_STRING
void SetInjectableArgvs(const std::vector< ::string>& new_argvs);
#endif  // GTEST_HAS_GLOBAL_STRING
void ClearInjectableArgvs();

#endif  // GTEST_HAS_DEATH_TEST

// Defines synchronization primitives.
#if GTEST_IS_THREADSAFE
# if GTEST_HAS_PTHREAD
// Sleeps for (roughly) n milliseconds.  This function is only for testing
// Google Test's own constructs.  Don't use it in user tests, either
// directly or indirectly.
inline void SleepMilliseconds(int n) {
  const timespec time = {
    0,                  // 0 seconds.
    n * 1000L * 1000L,  // And n ms.
  };
  nanosleep(&time, NULL);
}
# endif  // GTEST_HAS_PTHREAD

# if GTEST_HAS_NOTIFICATION_
// Notification has already been imported into the namespace.
// Nothing to do here.

# elif GTEST_HAS_PTHREAD
// Allows a controller thread to pause execution of newly created
// threads until notified.  Instances of this class must be created
// and destroyed in the controller thread.
//
// This class is only for testing Google Test's own constructs. Do not
// use it in user tests, either directly or indirectly.
class Notification {
 public:
  Notification() : notified_(false) {
    GTEST_CHECK_POSIX_SUCCESS_(pthread_mutex_init(&mutex_, NULL));
  }
  ~Notification() {
    pthread_mutex_destroy(&mutex_);
  }

  // Notifies all threads created with this notification to start. Must
  // be called from the controller thread.
  void Notify() {
    pthread_mutex_lock(&mutex_);
    notified_ = true;
    pthread_mutex_unlock(&mutex_);
  }

  // Blocks until the controller thread notifies. Must be called from a test
  // thread.
  void WaitForNotification() {
    for (;;) {
      pthread_mutex_lock(&mutex_);
      const bool notified = notified_;
      pthread_mutex_unlock(&mutex_);
      if (notified)
        break;
      SleepMilliseconds(10);
    }
  }

 private:
  pthread_mutex_t mutex_;
  bool notified_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(Notification);
};

# elif GTEST_OS_WINDOWS && !GTEST_OS_WINDOWS_PHONE && !GTEST_OS_WINDOWS_RT

GTEST_API_ void SleepMilliseconds(int n);

// Provides leak-safe Windows kernel handle ownership.
// Used in death tests and in threading support.
class GTEST_API_ AutoHandle {
 public:
  // Assume that Win32 HANDLE type is equivalent to void*. Doing so allows us to
  // avoid including <windows.h> in this header file. Including <windows.h> is
  // undesirable because it defines a lot of symbols and macros that tend to
  // conflict with client code. This assumption is verified by
  // WindowsTypesTest.HANDLEIsVoidStar.
  typedef void* Handle;
  AutoHandle();
  explicit AutoHandle(Handle handle);

  ~AutoHandle();

  Handle Get() const;
  void Reset();
  void Reset(Handle handle);

 private:
  // Returns true iff the handle is a valid handle object that can be closed.
  bool IsCloseable() const;

  Handle handle_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(AutoHandle);
};

// Allows a controller thread to pause execution of newly created
// threads until notified.  Instances of this class must be created
// and destroyed in the controller thread.
//
// This class is only for testing Google Test's own constructs. Do not
// use it in user tests, either directly or indirectly.
class GTEST_API_ Notification {
 public:
  Notification();
  void Notify();
  void WaitForNotification();

 private:
  AutoHandle event_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(Notification);
};
# endif  // GTEST_HAS_NOTIFICATION_

```

```
// On MinGW, we can have both GTEST_OS_WINDOWS and GTEST_HAS_PTHREAD
// defined, but we don’t want to use MinGW’s pthreads implementation, which
// has conformance problems with some versions of the POSIX standard.
# if GTEST_HAS_PTHREAD && !GTEST_OS_WINDOWS_MINGW

// As a C-function, ThreadFuncWithCLinkage cannot be templated itself.
// Consequently, it cannot select a correct instantiation of ThreadWithParam
// in order to call its Run(). Introducing ThreadWithParamBase as a
// non-templated base class for ThreadWithParam allows us to bypass this
// problem.
class ThreadWithParamBase {
 public:
  virtual ~ThreadWithParamBase() {}
  virtual void Run() = 0;
};

// pthread_create() accepts a pointer to a function type with the C linkage.
// According to the Standard (7.5/1), function types with different linkages
// are different even if they are otherwise identical.  Some compilers (for
// example, SunStudio) treat them as different types.  Since class methods
// cannot be defined with C-linkage we need to define a free C-function to
// pass into pthread_create().
extern "C" inline void* ThreadFuncWithCLinkage(void* thread) {
  static_cast<ThreadWithParamBase*>(thread)->Run();
  return NULL;
}

// Helper class for testing Google Test’s multi-threading constructs.
// To use it, write:
//
//   void ThreadFunc(int param) { /* Do things with param */ }
//   Notification thread_can_start;
//   ...
//   // The thread_can_start parameter is optional; you can supply NULL.
//   ThreadWithParam<int> thread(&ThreadFunc, 5, &thread_can_start);
//   thread_can_start.Notify();
//
// These classes are only for testing Google Test’s own constructs. Do
// not use them in user tests, either directly or indirectly.
template <typename T>
class ThreadWithParam : public ThreadWithParamBase {
 public:
  typedef void UserThreadFunc(T);

  ThreadWithParam(UserThreadFunc* func, T param, Notification* thread_can_start)
      : func_(func),
        param_(param),
        thread_can_start_(thread_can_start),
        finished_(false) {
    ThreadWithParamBase* const base = this;
    // The thread can be created only after all fields except thread_
    // have been initialized.
    GTEST_CHECK_POSIX_SUCCESS_(
        pthread_create(&thread_, 0, &ThreadFuncWithCLinkage, base));
  }
  ~ThreadWithParam() { Join(); }

  void Join() {
    if (!finished_) {
      GTEST_CHECK_POSIX_SUCCESS_(pthread_join(thread_, 0));
      finished_ = true;
    }
  }

  virtual void Run() {
    if (thread_can_start_ != NULL)
      thread_can_start_->WaitForNotification();
    func_(param_);
  }

 private:
  UserThreadFunc* const func_;  // User-supplied thread function.
  const T param_;  // User-supplied parameter to the thread function.
  // When non-NULL, used to block execution until the controller thread
  // notifies.
  Notification* const thread_can_start_;
  bool finished_;  // true iff we know that the thread function has finished.
  pthread_t thread_;  // The native thread object.

  GTEST_DISALLOW_COPY_AND_ASSIGN_(ThreadWithParam);
};
# endif  // !GTEST_OS_WINDOWS && GTEST_HAS_PTHREAD ||
         // GTEST_HAS_MUTEX_AND_THREAD_LOCAL_

# if GTEST_HAS_MUTEX_AND_THREAD_LOCAL_
// Mutex and ThreadLocal have already been imported into the namespace.
// Nothing to do here.
```

```

# elif GTEST_OS_WINDOWS && !GTEST_OS_WINDOWS_PHONE && !GTEST_OS_WINDOWS_RT

// Mutex implements mutex on Windows platforms.  It is used in conjunction
// with class MutexLock:
//
//   Mutex mutex;
//   ...
//   MutexLock lock(&mutex);  // Acquires the mutex and releases it at the
//                            // end of the current scope.
//
// A static Mutex *must* be defined or declared using one of the following
// macros:
//   GTEST_DEFINE_STATIC_MUTEX_(g_some_mutex);
//   GTEST_DECLARE_STATIC_MUTEX_(g_some_mutex);
//
// (A non-static Mutex is defined/declared in the usual way).
class GTEST_API_ Mutex {
 public:
  enum MutexType { kStatic = 0, kDynamic = 1 };
  // We rely on kStaticMutex being 0 as it is to what the linker initializes
  // type_ in static mutexes.  critical_section_ will be initialized lazily
  // in ThreadSafeLazyInit().
  enum StaticConstructorSelector { kStaticMutex = 0 };

  // This constructor intentionally does nothing.  It relies on type_ being
  // statically initialized to 0 (effectively setting it to kStatic) and on
  // ThreadSafeLazyInit() to lazily initialize the rest of the members.
  explicit Mutex(StaticConstructorSelector /*dummy*/) {}

  Mutex();
  ~Mutex();

  void Lock();

  void Unlock();

  // Does nothing if the current thread holds the mutex. Otherwise, crashes
  // with high probability.
  void AssertHeld();

 private:
  // Initializes owner_thread_id_ and critical_section_ in static mutexes.
  void ThreadSafeLazyInit();

  // Per https://blogs.msdn.microsoft.com/oldnewthing/20040223-00/?p=40503,
  // we assume that 0 is an invalid value for thread IDs.
  unsigned int owner_thread_id_;

  // For static mutexes, we rely on these members being initialized to zeros
  // by the linker.
  MutexType type_;
  long critical_section_init_phase_;  // NOLINT
  GTEST_CRITICAL_SECTION* critical_section_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(Mutex);
};

# define GTEST_DECLARE_STATIC_MUTEX_(mutex) \
    extern ::testing::internal::Mutex mutex

# define GTEST_DEFINE_STATIC_MUTEX_(mutex) \
    ::testing::internal::Mutex mutex(::testing::internal::Mutex::kStaticMutex)

// We cannot name this class MutexLock because the ctor declaration would
// conflict with a macro named MutexLock, which is defined on some
// platforms. That macro is used as a defensive measure to prevent against
// inadvertent misuses of MutexLock like "MutexLock(&mu)" rather than
// "MutexLock l(&mu)".  Hence the typedef trick below.
class GTestMutexLock {
 public:
  explicit GTestMutexLock(Mutex* mutex)
      : mutex_(mutex) { mutex_->Lock(); }

  ~GTestMutexLock() { mutex_->Unlock(); }

 private:
  Mutex* const mutex_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(GTestMutexLock);
};

typedef GTestMutexLock MutexLock;

// Base class for ValueHolder<T>.  Allows a caller to hold and delete a value
// without knowing its type.
class ThreadLocalValueHolderBase {
 public:
  virtual ~ThreadLocalValueHolderBase() {}
};

// Provides a way for a thread to send notifications to a ThreadLocal
// regardless of its parameter type.
class ThreadLocalBase {
 public:
  // Creates a new ValueHolder<T> object holding a default value passed to
  // this ThreadLocal<T>'s constructor and returns it.  It is the caller's
  // responsibility not to call this when the ThreadLocal<T> instance already
  // has a value on the current thread.
  virtual ThreadLocalValueHolderBase* NewValueForCurrentThread() const = 0;

 protected:
  ThreadLocalBase() {}
  virtual ~ThreadLocalBase() {}

 private:
  GTEST_DISALLOW_COPY_AND_ASSIGN_(ThreadLocalBase);
};

// Maps a thread to a set of ThreadLocals that have values instantiated on that
// thread and notifies them when the thread exits.  A ThreadLocal instance is
// expected to persist until all threads it has values on have terminated.
class GTEST_API_ ThreadLocalRegistry {
 public:
  // Registers thread_local_instance as having value on the current thread.
  // Returns a value that can be used to identify the thread from other threads.
  static ThreadLocalValueHolderBase* GetValueOnCurrentThread(
      const ThreadLocalBase* thread_local_instance);

  // Invoked when a ThreadLocal instance is destroyed.
  static void OnThreadLocalDestroyed(
      const ThreadLocalBase* thread_local_instance);
};

class GTEST_API_ ThreadWithParamBase {
 public:
  void Join();

 protected:
  class Runnable {
   public:
    virtual ~Runnable() {}
    virtual void Run() = 0;
  };

  ThreadWithParamBase(Runnable *runnable, Notification* thread_can_start);
  virtual ~ThreadWithParamBase();

 private:
  AutoHandle thread_;
};

// Helper class for testing Google Test's multi-threading constructs.
template <typename T>
class ThreadWithParam : public ThreadWithParamBase {
 public:
  typedef void UserThreadFunc(T);

  ThreadWithParam(UserThreadFunc* func, T param, Notification* thread_can_start)
      : ThreadWithParamBase(new RunnableImpl(func, param), thread_can_start) {
  }
  virtual ~ThreadWithParam() {}

 private:
  class RunnableImpl : public Runnable {
   public:
    RunnableImpl(UserThreadFunc* func, T param)
        : func_(func),
          param_(param) {
    }
    virtual ~RunnableImpl() {}
    virtual void Run() {
      func_(param_);
    }

   private:
    UserThreadFunc* const func_;
    const T param_;

    GTEST_DISALLOW_COPY_AND_ASSIGN_(RunnableImpl);
  };

  GTEST_DISALLOW_COPY_AND_ASSIGN_(ThreadWithParam);
};
```

```

// Implements thread-local storage on Windows systems.
//
//   // Thread 1
//   ThreadLocal<int> tl(100);  // 100 is the default value for each thread.
//
//   // Thread 2
//   tl.set(150);  // Changes the value for thread 2 only.
//   EXPECT_EQ(150, tl.get());
//
//   // Thread 1
//   EXPECT_EQ(100, tl.get());  // In thread 1, tl has the original value.
//   tl.set(200);
//   EXPECT_EQ(200, tl.get());
//
// The template type argument T must have a public copy constructor.
// In addition, the default ThreadLocal constructor requires T to have
// a public default constructor.
//
// The users of a TheadLocal instance have to make sure that all but one
// threads (including the main one) using that instance have exited before
// destroying it. Otherwise, the per-thread objects managed for them by the
// ThreadLocal instance are not guaranteed to be destroyed on all platforms.
//
// Google Test only uses global ThreadLocal objects.  That means they
// will die after main() has returned.  Therefore, no per-thread
// object managed by Google Test will be leaked as long as all threads
// using Google Test have exited when main() returns.
template <typename T>
class ThreadLocal : public ThreadLocalBase {
 public:
  ThreadLocal() : default_factory_(new DefaultValueHolderFactory()) {}
  explicit ThreadLocal(const T& value)
      : default_factory_(new InstanceValueHolderFactory(value)) {}

  ~ThreadLocal() { ThreadLocalRegistry::OnThreadLocalDestroyed(this); }

  T* pointer() { return GetOrCreateValue(); }
  const T* pointer() const { return GetOrCreateValue(); }
  const T& get() const { return *pointer(); }
  void set(const T& value) { *pointer() = value; }

 private:
  // Holds a value of T.  Can be deleted via its base class without the caller
  // knowing the type of T.
  class ValueHolder : public ThreadLocalValueHolderBase {
   public:
    ValueHolder() : value_() {}
    explicit ValueHolder(const T& value) : value_(value) {}

    T* pointer() { return &value_; }

   private:
    T value_;
    GTEST_DISALLOW_COPY_AND_ASSIGN_(ValueHolder);
  };


  T* GetOrCreateValue() const {
    return static_cast<ValueHolder*>(
        ThreadLocalRegistry::GetValueOnCurrentThread(this))->pointer();
  }

  virtual ThreadLocalValueHolderBase* NewValueForCurrentThread() const {
    return default_factory_->MakeNewHolder();
  }

  class ValueHolderFactory {
   public:
    ValueHolderFactory() {}
    virtual ~ValueHolderFactory() {}
    virtual ValueHolder* MakeNewHolder() const = 0;

   private:
    GTEST_DISALLOW_COPY_AND_ASSIGN_(ValueHolderFactory);
  };

  class DefaultValueHolderFactory : public ValueHolderFactory {
   public:
    DefaultValueHolderFactory() {}
    virtual ValueHolder* MakeNewHolder() const { return new ValueHolder(); }

   private:
    GTEST_DISALLOW_COPY_AND_ASSIGN_(DefaultValueHolderFactory);
  };

  class InstanceValueHolderFactory : public ValueHolderFactory {
   public:
    explicit InstanceValueHolderFactory(const T& value) : value_(value) {}
    virtual ValueHolder* MakeNewHolder() const {
      return new ValueHolder(value_);
    }

   private:
    const T value_;  // The value for each thread.

    GTEST_DISALLOW_COPY_AND_ASSIGN_(InstanceValueHolderFactory);
  };

  scoped_ptr<ValueHolderFactory> default_factory_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(ThreadLocal);
};

# elif GTEST_HAS_PTHREAD

// MutexBase and Mutex implement mutex on pthreads-based platforms.
class MutexBase {
 public:
  // Acquires this mutex.
  void Lock() {
    GTEST_CHECK_POSIX_SUCCESS_(pthread_mutex_lock(&mutex_));
    owner_ = pthread_self();
    has_owner_ = true;
  }

  // Releases this mutex.
  void Unlock() {
    // Since the lock is being released the owner_ field should no longer be
    // considered valid. We don’t protect writing to has_owner_ here, as it’s
    // the caller’s responsibility to ensure that the current thread holds the
    // mutex when this is called.
    has_owner_ = false;
    GTEST_CHECK_POSIX_SUCCESS_(pthread_mutex_unlock(&mutex_));
  }

  // Does nothing if the current thread holds the mutex. Otherwise, crashes
  // with high probability.
  void AssertHeld() const {
    GTEST_CHECK_(has_owner_ && pthread_equal(owner_, pthread_self()))
        « "The current thread is not holding the mutex @" « this;
  }

  // A static mutex may be used before main() is entered.  It may even
  // be used before the dynamic initialization stage.  Therefore we
  // must be able to initialize a static mutex object at link time.
  // This means MutexBase has to be a POD and its member variables
  // have to be public.
 public:
  pthread_mutex_t mutex_;  // The underlying pthread mutex.
  // has_owner_ indicates whether the owner_ field below contains a valid thread
  // ID and is therefore safe to inspect (e.g., to use in pthread_equal()). All
  // accesses to the owner_ field should be protected by a check of this field.
  // An alternative might be to memset() owner_ to all zeros, but there’s no
  // guarantee that a zero’d pthread_t is necessarily invalid or even different
  // from pthread_self().
  bool has_owner_;
  pthread_t owner_;  // The thread holding the mutex.
};

// Forward-declares a static mutex.
#  define GTEST_DECLARE_STATIC_MUTEX_(mutex) \
     extern ::testing::internal::MutexBase mutex

// Defines and statically (i.e. at link time) initializes a static mutex.
// The initialization list here does not explicitly initialize each field,
// instead relying on default initialization for the unspecified fields. In
// particular, the owner_ field (a pthread_t) is not explicitly initialized.
// This allows initialization to work whether pthread_t is a scalar or struct.
// The flag -Wmissing-field-initializers must not be specified for this to work.
#define GTEST_DEFINE_STATIC_MUTEX_(mutex) \
  ::testing::internal::MutexBase mutex = {PTHREAD_MUTEX_INITIALIZER, false, 0}

// The Mutex class can only be used for mutexes created at runtime. It
// shares its API with MutexBase otherwise.
class Mutex : public MutexBase {
 public:
  Mutex() {
    GTEST_CHECK_POSIX_SUCCESS_(pthread_mutex_init(&mutex_, NULL));
    has_owner_ = false;
  }
  ~Mutex() {
    GTEST_CHECK_POSIX_SUCCESS_(pthread_mutex_destroy(&mutex_));
  }

 private:
  GTEST_DISALLOW_COPY_AND_ASSIGN_(Mutex);
};

// We cannot name this class MutexLock because the ctor declaration would
// conflict with a macro named MutexLock, which is defined on some
// platforms. That macro is used as a defensive measure to prevent against
// inadvertent misuses of MutexLock like "MutexLock(&mu)" rather than
// "MutexLock l(&mu)".  Hence the typedef trick below.
class GTestMutexLock {
 public:
  explicit GTestMutexLock(MutexBase* mutex)
      : mutex_(mutex) { mutex_->Lock(); }

  ~GTestMutexLock() { mutex_->Unlock(); }

 private:
  MutexBase* const mutex_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(GTestMutexLock);
};

typedef GTestMutexLock MutexLock;

// Helpers for ThreadLocal.

// pthread_key_create() requires DeleteThreadLocalValue() to have
// C-linkage.  Therefore it cannot be templatized to access
// ThreadLocal<T>.  Hence the need for class
// ThreadLocalValueHolderBase.
class ThreadLocalValueHolderBase {
 public:
  virtual ~ThreadLocalValueHolderBase() {}
};

// Called by pthread to delete thread-local data stored by
// pthread_setspecific().
extern "C" inline void DeleteThreadLocalValue(void* value_holder) {
  delete static_cast<ThreadLocalValueHolderBase*>(value_holder);
}

// Implements thread-local storage on pthreads-based systems.
template <typename T>
class GTEST_API_ ThreadLocal {
 public:
  ThreadLocal()
      : key_(CreateKey()), default_factory_(new DefaultValueHolderFactory()) {}
  explicit ThreadLocal(const T& value)
      : key_(CreateKey()),
        default_factory_(new InstanceValueHolderFactory(value)) {}

  ~ThreadLocal() {
    // Destroys the managed object for the current thread, if any.
    DeleteThreadLocalValue(pthread_getspecific(key_));

    // Releases resources associated with the key.  This will *not*
    // delete managed objects for other threads.
    GTEST_CHECK_POSIX_SUCCESS_(pthread_key_delete(key_));
  }

  T* pointer() { return GetOrCreateValue(); }
  const T* pointer() const { return GetOrCreateValue(); }
  const T& get() const { return *pointer(); }
  void set(const T& value) { *pointer() = value; }

 private:
  // Holds a value of type T.
  class ValueHolder : public ThreadLocalValueHolderBase {
   public:
    ValueHolder() : value_() {}
    explicit ValueHolder(const T& value) : value_(value) {}

    T* pointer() { return &value_; }

   private:
    T value_;
    GTEST_DISALLOW_COPY_AND_ASSIGN_(ValueHolder);
  };

  static pthread_key_t CreateKey() {
    pthread_key_t key;
    // When a thread exits, DeleteThreadLocalValue() will be called on
    // the object managed for that thread.
    GTEST_CHECK_POSIX_SUCCESS_(
        pthread_key_create(&key, &DeleteThreadLocalValue));
    return key;
  }

  T* GetOrCreateValue() const {
    ThreadLocalValueHolderBase* const holder =
        static_cast<ThreadLocalValueHolderBase*>(pthread_getspecific(key_));
    if (holder != NULL) {
      return CheckedDowncastToActualType<ValueHolder>(holder)->pointer();
    }

    ValueHolder* const new_holder = default_factory_->MakeNewHolder();
    ThreadLocalValueHolderBase* const holder_base = new_holder;
    GTEST_CHECK_POSIX_SUCCESS_(pthread_setspecific(key_, holder_base));
    return new_holder->pointer();
  }

  class ValueHolderFactory {
   public:
    ValueHolderFactory() {}
    virtual ~ValueHolderFactory() {}
    virtual ValueHolder* MakeNewHolder() const = 0;

   private:
    GTEST_DISALLOW_COPY_AND_ASSIGN_(ValueHolderFactory);
  };

  class DefaultValueHolderFactory : public ValueHolderFactory {
   public:
    DefaultValueHolderFactory() {}
    virtual ValueHolder* MakeNewHolder() const { return new ValueHolder(); }

   private:
    GTEST_DISALLOW_COPY_AND_ASSIGN_(DefaultValueHolderFactory);
  };

  class InstanceValueHolderFactory : public ValueHolderFactory {
   public:
    explicit InstanceValueHolderFactory(const T& value) : value_(value) {}
    virtual ValueHolder* MakeNewHolder() const {
      return new ValueHolder(value_);
    }

   private:
    const T value_;  // The value for each thread.

    GTEST_DISALLOW_COPY_AND_ASSIGN_(InstanceValueHolderFactory);
  };

  // A key pthreads uses for looking up per-thread values.
  const pthread_key_t key_;
  scoped_ptr<ValueHolderFactory> default_factory_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(ThreadLocal);
};

# endif  // GTEST_HAS_MUTEX_AND_THREAD_LOCAL_

#else  // GTEST_IS_THREADSAFE

// A dummy implementation of synchronization primitives (mutex, lock,
// and thread-local variable).  Necessary for compiling Google Test where
// mutex is not supported - using Google Test in multiple threads is not
// supported on such platforms.

class Mutex {
 public:
  Mutex() {}
  void Lock() {}
  void Unlock() {}
  void AssertHeld() const {}
};

# define GTEST_DECLARE_STATIC_MUTEX_(mutex) \
  extern ::testing::internal::Mutex mutex

# define GTEST_DEFINE_STATIC_MUTEX_(mutex) ::testing::internal::Mutex mutex

// We cannot name this class MutexLock because the ctor declaration would
// conflict with a macro named MutexLock, which is defined on some
// platforms. That macro is used as a defensive measure to prevent against
// inadvertent misuses of MutexLock like "MutexLock(&mu)" rather than
// "MutexLock l(&mu)".  Hence the typedef trick below.
class GTestMutexLock {
 public:
  explicit GTestMutexLock(Mutex*) {}  // NOLINT
};

typedef GTestMutexLock MutexLock;
```

```

template <typename T>
class GTEST_API_ ThreadLocal {
 public:
  ThreadLocal() : value_() {}
  explicit ThreadLocal(const T& value) : value_(value) {}
  T* pointer() { return &value_; }
  const T* pointer() const { return &value_; }
  const T& get() const { return value_; }
  void set(const T& value) { value_ = value; }
 private:
  T value_;
};

#endif  // GTEST_IS_THREADSAFE

// Returns the number of threads running in the process, or 0 to indicate that
// we cannot detect it.
GTEST_API_ size_t GetThreadCount();

// Passing non-POD classes through ellipsis (...) crashes the ARM
// compiler and generates a warning in Sun Studio before 12u4. The Nokia Symbian
// and the IBM XL C/C++ compiler try to instantiate a copy constructor
// for objects passed through ellipsis (...), failing for uncopyable
// objects.  We define this to ensure that only POD is passed through
// ellipsis on these systems.
#if defined(__SYMBIAN32__) || defined(__IBMCPP__) || \
     (defined(__SUNPRO_CC) && __SUNPRO_CC < 0x5130)
// We lose support for NULL detection where the compiler doesn't like
// passing non-POD classes through ellipsis (...).
# define GTEST_ELLIPSIS_NEEDS_POD_ 1
#else
# define GTEST_CAN_COMPARE_NULL 1
#endif

// The Nokia Symbian and IBM XL C/C++ compilers cannot decide between
// const T& and const T* in a function template.  These compilers
// _can_ decide between class template specializations for T and T*,
// so a tr1::type_traits-like is_pointer works.
#if defined(__SYMBIAN32__) || defined(__IBMCPP__)
# define GTEST_NEEDS_IS_POINTER_ 1
#endif

template <bool bool_value>
struct bool_constant {
  typedef bool_constant<bool_value> type;
  static const bool value = bool_value;
};
template <bool bool_value> const bool bool_constant<bool_value>::value;

typedef bool_constant<false> false_type;
typedef bool_constant<true> true_type;

template <typename T, typename U>
struct is_same : public false_type {};

template <typename T>
struct is_same<T, T> : public true_type {};


template <typename T>
struct is_pointer : public false_type {};

template <typename T>
struct is_pointer<T*> : public true_type {};

template <typename Iterator>
struct IteratorTraits {
  typedef typename Iterator::value_type value_type;
};


template <typename T>
struct IteratorTraits<T*> {
  typedef T value_type;
};

template <typename T>
struct IteratorTraits<const T*> {
  typedef T value_type;
};

#if GTEST_OS_WINDOWS
# define GTEST_PATH_SEP_ "\\"
# define GTEST_HAS_ALT_PATH_SEP_ 1
// The biggest signed integer type the compiler supports.
typedef __int64 BiggestInt;
```

```
#else
# define GTEST_PATH_SEP_ "/"
# define GTEST_HAS_ALT_PATH_SEP_ 0
typedef long long BiggestInt;  // NOLINT
#endif  // GTEST_OS_WINDOWS

// Utilities for char.

// isspace(int ch) and friends accept an unsigned char or EOF.  char
// may be signed, depending on the compiler (or compiler flags).
// Therefore we need to cast a char to unsigned char before calling
// isspace(), etc.

inline bool IsAlpha(char ch) {
  return isalpha(static_cast<unsigned char>(ch)) != 0;
}
inline bool IsAlNum(char ch) {
  return isalnum(static_cast<unsigned char>(ch)) != 0;
}
inline bool IsDigit(char ch) {
  return isdigit(static_cast<unsigned char>(ch)) != 0;
}
inline bool IsLower(char ch) {
  return islower(static_cast<unsigned char>(ch)) != 0;
}
inline bool IsSpace(char ch) {
  return isspace(static_cast<unsigned char>(ch)) != 0;
}
inline bool IsUpper(char ch) {
  return isupper(static_cast<unsigned char>(ch)) != 0;
}
inline bool IsXDigit(char ch) {
  return isxdigit(static_cast<unsigned char>(ch)) != 0;
}
inline bool IsXDigit(wchar_t ch) {
  const unsigned char low_byte = static_cast<unsigned char>(ch);
  return ch == low_byte && isxdigit(low_byte) != 0;
}

inline char ToLower(char ch) {
  return static_cast<char>(tolower(static_cast<unsigned char>(ch)));
}
inline char ToUpper(char ch) {
  return static_cast<char>(toupper(static_cast<unsigned char>(ch)));
}

inline std::string StripTrailingSpaces(std::string str) {
  std::string::iterator it = str.end();
  while (it != str.begin() && IsSpace(*--it))
    it = str.erase(it);
  return str;
}

// The testing::internal::posix namespace holds wrappers for common
// POSIX functions.  These wrappers hide the differences between
// Windows/MSVC and POSIX systems.  Since some compilers define these
// standard functions as macros, the wrapper cannot have the same name
// as the wrapped function.

namespace posix {

// Functions with a different name on Windows.

#if GTEST_OS_WINDOWS

typedef struct _stat StatStruct;

# ifdef __BORLANDC__
inline int IsATTY(int fd) { return isatty(fd); }
inline int StrCaseCmp(const char* s1, const char* s2) {
  return stricmp(s1, s2);
}
inline char* StrDup(const char* src) { return strdup(src); }
# else  // !__BORLANDC__
#  if GTEST_OS_WINDOWS_MOBILE
inline int IsATTY(int /* fd */) { return 0; }
#  else
inline int IsATTY(int fd) { return _isatty(fd); }
#  endif  // GTEST_OS_WINDOWS_MOBILE
inline int StrCaseCmp(const char* s1, const char* s2) {
  return _stricmp(s1, s2);
}
inline char* StrDup(const char* src) { return _strdup(src); }
# endif  // __BORLANDC__

# if GTEST_OS_WINDOWS_MOBILE
inline int FileNo(FILE* file) { return reinterpret_cast<int>(_fileno(file)); }
```

```
// Stat(), RmDir(), and IsDir() are not needed on Windows CE at this
// time and thus not defined there.
# else
inline int FileNo(FILE* file) { return _fileno(file); }
inline int Stat(const char* path, StatStruct* buf) { return _stat(path, buf); }
inline int RmDir(const char* dir) { return _rmdir(dir); }
inline bool IsDir(const StatStruct& st) {
  return (_S_IFDIR & st.st_mode) != 0;
}
# endif  // GTEST_OS_WINDOWS_MOBILE

#else

typedef struct stat StatStruct;

inline int FileNo(FILE* file) { return fileno(file); }
inline int IsATTY(int fd) { return isatty(fd); }
inline int Stat(const char* path, StatStruct* buf) { return stat(path, buf); }
inline int StrCaseCmp(const char* s1, const char* s2) {
  return strcasecmp(s1, s2);
}
inline char* StrDup(const char* src) { return strdup(src); }
inline int RmDir(const char* dir) { return rmdir(dir); }
inline bool IsDir(const StatStruct& st) { return S_ISDIR(st.st_mode); }

#endif  // GTEST_OS_WINDOWS

// Functions deprecated by MSVC 8.0.

GTEST_DISABLE_MSC_DEPRECATED_PUSH_()

inline const char* StrNCpy(char* dest, const char* src, size_t n) {
  return strncpy(dest, src, n);
}

// ChDir(), FReopen(), FDOpen(), Read(), Write(), Close(), and
// StrError() aren't needed on Windows CE at this time and thus not
// defined there.

#if !GTEST_OS_WINDOWS_MOBILE && !GTEST_OS_WINDOWS_PHONE && !GTEST_OS_WINDOWS_RT
inline int ChDir(const char* dir) { return chdir(dir); }
#endif
inline FILE* FOpen(const char* path, const char* mode) {
  return fopen(path, mode);
}
#if !GTEST_OS_WINDOWS_MOBILE
inline FILE *FReopen(const char* path, const char* mode, FILE* stream) {
  return freopen(path, mode, stream);
}
inline FILE* FDOpen(int fd, const char* mode) { return fdopen(fd, mode); }
#endif
inline int FClose(FILE* fp) { return fclose(fp); }
#if !GTEST_OS_WINDOWS_MOBILE
inline int Read(int fd, void* buf, unsigned int count) {
  return static_cast<int>(read(fd, buf, count));
}
inline int Write(int fd, const void* buf, unsigned int count) {
  return static_cast<int>(write(fd, buf, count));
}
inline int Close(int fd) { return close(fd); }
inline const char* StrError(int errnum) { return strerror(errnum); }
#endif
inline const char* GetEnv(const char* name) {
#if GTEST_OS_WINDOWS_MOBILE || GTEST_OS_WINDOWS_PHONE || GTEST_OS_WINDOWS_RT
  // We are on Windows CE, which has no environment variables.
  static_cast<void>(name);  // To prevent 'unused argument' warning.
  return NULL;
#elif defined(__BORLANDC__) || defined(__SunOS_5_8) || defined(__SunOS_5_9)
  // Environment variables which we programmatically clear will be set to the
  // empty string rather than unset (NULL).  Handle that case.
  const char* const env = getenv(name);
  return (env != NULL && env[0] != '\0') ? env : NULL;
#else
  return getenv(name);
#endif
}

GTEST_DISABLE_MSC_DEPRECATED_POP_()

#if GTEST_OS_WINDOWS_MOBILE
// Windows CE has no C library. The abort() function is used in
// several places in Google Test. This implementation provides a reasonable
// imitation of standard behaviour.
void Abort();
#else
inline void Abort() { abort(); }
#endif  // GTEST_OS_WINDOWS_MOBILE
```

```

}  // namespace posix

// MSVC "deprecates" snprintf and issues warnings wherever it is used.  In
// order to avoid these warnings, we need to use _snprintf or _snprintf_s on
// MSVC-based platforms.  We map the GTEST_SNPRINTF_ macro to the appropriate
// function in order to achieve that.  We use macro definition here because
// snprintf is a variadic function.
#if _MSC_VER >= 1400 && !GTEST_OS_WINDOWS_MOBILE
// MSVC 2005 and above support variadic macros.
# define GTEST_SNPRINTF_(buffer, size, format, ...) \
     _snprintf_s(buffer, size, size, format, __VA_ARGS__)
#elif defined(_MSC_VER)
// Windows CE does not define _snprintf_s and MSVC prior to 2005 doesn't
// complain about _snprintf.
# define GTEST_SNPRINTF_ _snprintf
#else
# define GTEST_SNPRINTF_ snprintf
#endif

// The maximum number a BiggestInt can represent.  This definition
// works no matter BiggestInt is represented in one's complement or
// two's complement.
//
// We cannot rely on numeric_limits in STL, as __int64 and long long
// are not part of standard C++ and numeric_limits doesn't need to be
// defined for them.
const BiggestInt kMaxBiggestInt =
    ~(static_cast<BiggestInt>(1) « (8*sizeof(BiggestInt) - 1));

// This template class serves as a compile-time function from size to
// type.  It maps a size in bytes to a primitive type with that
// size. e.g.
//
//   TypeWithSize<4>::UInt
//
// is typedef-ed to be unsigned int (unsigned integer made up of 4
// bytes).
//
// Such functionality should belong to STL, but I cannot find it
// there.
//
// Google Test uses this class in the implementation of floating-point
// comparison.
//
// For now it only handles UInt (unsigned int) as that's all Google Test
// needs.  Other types can be easily added in the future if need
// arises.
template <size_t size>
class TypeWithSize {
 public:
  // This prevents the user from using TypeWithSize<N> with incorrect
  // values of N.
  typedef void UInt;
};

// The specialization for size 4.
template <>
class TypeWithSize<4> {
 public:
  // unsigned int has size 4 in both gcc and MSVC.
  //
  // As base/basictypes.h doesn't compile on Windows, we cannot use
  // uint32, uint64, and etc here.
  typedef int Int;
  typedef unsigned int UInt;
};

// The specialization for size 8.
template <>
class TypeWithSize<8> {
 public:
#if GTEST_OS_WINDOWS
  typedef __int64 Int;
  typedef unsigned __int64 UInt;
#else
  typedef long long Int;  // NOLINT
  typedef unsigned long long UInt;  // NOLINT
#endif  // GTEST_OS_WINDOWS
};

// Integer types of known sizes.
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;  // Represents time in milliseconds.
```

```

// Utilities for command line flags and environment variables.

// Macro for referencing flags.
#if !defined(GTEST_FLAG)
# define GTEST_FLAG(name) FLAGS_gtest_##name
#endif  // !defined(GTEST_FLAG)

#if !defined(GTEST_USE_OWN_FLAGFILE_FLAG_)
# define GTEST_USE_OWN_FLAGFILE_FLAG_ 1
#endif  // !defined(GTEST_USE_OWN_FLAGFILE_FLAG_)

#if !defined(GTEST_DECLARE_bool_)
# define GTEST_FLAG_SAVER_ ::testing::internal::GTestFlagSaver

// Macros for declaring flags.
# define GTEST_DECLARE_bool_(name) GTEST_API_ extern bool GTEST_FLAG(name)
# define GTEST_DECLARE_int32_(name) \
    GTEST_API_ extern ::testing::internal::Int32 GTEST_FLAG(name)
# define GTEST_DECLARE_string_(name) \
    GTEST_API_ extern ::std::string GTEST_FLAG(name)

// Macros for defining flags.
# define GTEST_DEFINE_bool_(name, default_val, doc) \
    GTEST_API_ bool GTEST_FLAG(name) = (default_val)
# define GTEST_DEFINE_int32_(name, default_val, doc) \
    GTEST_API_ ::testing::internal::Int32 GTEST_FLAG(name) = (default_val)
# define GTEST_DEFINE_string_(name, default_val, doc) \
    GTEST_API_ ::std::string GTEST_FLAG(name) = (default_val)

#endif  // !defined(GTEST_DECLARE_bool_)

// Thread annotations
#if !defined(GTEST_EXCLUSIVE_LOCK_REQUIRED_)
# define GTEST_EXCLUSIVE_LOCK_REQUIRED_(locks)
# define GTEST_LOCK_EXCLUDED_(locks)
#endif  // !defined(GTEST_EXCLUSIVE_LOCK_REQUIRED_)

// Parses 'str' for a 32-bit signed integer.  If successful, writes the result
// to *value and returns true; otherwise leaves *value unchanged and returns
// false.
// FIXME: Find a better way to refactor flag and environment parsing
// out of both gtest-port.cc and gtest.cc to avoid exporting this utility
// function.
bool ParseInt32(const Message& src_text, const char* str, Int32* value);

// Parses a bool/Int32/string from the environment variable
// corresponding to the given Google Test flag.
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);

}  // namespace internal
}  // namespace testing

#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.

```
// Copyright 2005, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// The Google C++ Testing and Mocking Framework (Google Test)
//
// This header file defines internal utilities needed for implementing
// death tests.  They are subject to change without notice.
// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_DEATH_TEST_INTERNAL_H_
#define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_DEATH_TEST_INTERNAL_H_

#include "gtest/internal/gtest-internal.h"

#include <stdio.h>

namespace testing {
namespace internal {

GTEST_DECLARE_string_(internal_run_death_test);

// Names of the flags (needed for parsing Google Test flags).
const char kDeathTestStyleFlag[] = "death_test_style";
const char kDeathTestUseFork[] = "death_test_use_fork";
const char kInternalRunDeathTestFlag[] = "internal_run_death_test";

```

```
#if GTEST_HAS_DEATH_TEST

GTEST_DISABLE_MSC_WARNINGS_PUSH_(4251 \
/* class A needs to have dll-interface to be used by clients of class B */)

// DeathTest is a class that hides much of the complexity of the
// GTEST_DEATH_TEST_ macro.  It is abstract; its static Create method
// returns a concrete class that depends on the prevailing death test
// style, as defined by the --gtest_death_test_style and/or
// --gtest_internal_run_death_test flags.

// In describing the results of death tests, these terms are used with
// the corresponding definitions:
//
// exit status:  The integer exit information in the format specified
//               by wait(2)
// exit code:    The integer code passed to exit(3), _exit(2), or
//               returned from main()
class GTEST_API_ DeathTest {
 public:
  // Create returns false if there was an error determining the
  // appropriate action to take for the current death test; for example,
  // if the gtest_death_test_style flag is set to an invalid value.
  // The LastMessage method will return a more detailed message in that
  // case.  Otherwise, the DeathTest pointer pointed to by the "test"
  // argument is set.  If the death test should be skipped, the pointer
  // is set to NULL; otherwise, it is set to the address of a new concrete
  // DeathTest object that controls the execution of the current test.
  static bool Create(const char* statement, const RE* regex,
                     const char* file, int line, DeathTest** test);
  DeathTest();
  virtual ~DeathTest() { }

  // A helper class that aborts a death test when it's deleted.
  class ReturnSentinel {
   public:
    explicit ReturnSentinel(DeathTest* test) : test_(test) { }
    ~ReturnSentinel() { test_->Abort(TEST_ENCOUNTERED_RETURN_STATEMENT); }
   private:
    DeathTest* const test_;
    GTEST_DISALLOW_COPY_AND_ASSIGN_(ReturnSentinel);
  } GTEST_ATTRIBUTE_UNUSED_;

  // An enumeration of possible roles that may be taken when a death
  // test is encountered.  EXECUTE means that the death test logic should
  // be executed immediately.  OVERSEE means that the program should prepare
  // the appropriate environment for a child process to execute the death
  // test, then wait for it to complete.
  enum TestRole { OVERSEE_TEST, EXECUTE_TEST };

  // An enumeration of the three reasons that a test might be aborted.
  enum AbortReason {
    TEST_ENCOUNTERED_RETURN_STATEMENT,
    TEST_THREW_EXCEPTION,
    TEST_DID_NOT_DIE
  };

  // Assumes one of the above roles.
  virtual TestRole AssumeRole() = 0;

  // Waits for the death test to finish and returns its status.
  virtual int Wait() = 0;

  // Returns true if the death test passed; that is, the test process
  // exited during the test, its exit status matches a user-supplied
  // predicate, and its stderr output matches a user-supplied regular
  // expression.
  // The user-supplied predicate may be a macro expression rather
  // than a function pointer or functor, or else Wait and Passed could
  // be combined.
  virtual bool Passed(bool exit_status_ok) = 0;

  // Signals that the death test did not die as expected.
  virtual void Abort(AbortReason reason) = 0;

  // Returns a human-readable outcome message regarding the outcome of
  // the last death test.
  static const char* LastMessage();

  static void set_last_death_test_message(const std::string& message);

 private:
  // A string containing a description of the outcome of the last death test.
  static std::string last_death_test_message_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(DeathTest);
};
```

```

GTEST_DISABLE_MSC_WARNINGS_POP_()  //  4251

// Factory interface for death tests.  May be mocked out for testing.
class DeathTestFactory {
 public:
  virtual ~DeathTestFactory() { }
  virtual bool Create(const char* statement, const RE* regex,
                      const char* file, int line, DeathTest** test) = 0;
};

// A concrete DeathTestFactory implementation for normal use.
class DefaultDeathTestFactory : public DeathTestFactory {
 public:
  virtual bool Create(const char* statement, const RE* regex,
                      const char* file, int line, DeathTest** test);
};

// Returns true if exit_status describes a process that was terminated
// by a signal, or exited normally with a nonzero exit code.
GTEST_API_ bool ExitedUnsuccessfully(int exit_status);

// Traps C++ exceptions escaping statement and reports them as test
// failures. Note that trapping SEH exceptions is not implemented here.
# if GTEST_HAS_EXCEPTIONS
#  define GTEST_EXECUTE_DEATH_TEST_STATEMENT_(statement, death_test) \
  try { \
    GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement); \
  } catch (const ::std::exception& gtest_exception) { \
    fprintf(\
        stderr, \
        "\n%s: Caught std::exception-derived exception escaping the " \
        "death test statement. Exception message: %s\n", \
        ::testing::internal::FormatFileLocation(__FILE__, __LINE__).c_str(), \
        gtest_exception.what()); \
    fflush(stderr); \
    death_test->Abort(::testing::internal::DeathTest::TEST_THREW_EXCEPTION); \
  } catch (...) { \
    death_test->Abort(::testing::internal::DeathTest::TEST_THREW_EXCEPTION); \
  }

# else
#  define GTEST_EXECUTE_DEATH_TEST_STATEMENT_(statement, death_test) \
  GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement)

# endif

// This macro is for implementing ASSERT_DEATH*, EXPECT_DEATH*,
// ASSERT_EXIT*, and EXPECT_EXIT*.
# define GTEST_DEATH_TEST_(statement, predicate, regex, fail) \
  GTEST_AMBIGUOUS_ELSE_BLOCKER_ \
  if (::testing::internal::AlwaysTrue()) { \
    const ::testing::internal::RE& gtest_regex = (regex); \
    ::testing::internal::DeathTest* gtest_dt; \
    if (!::testing::internal::DeathTest::Create(#statement, &gtest_regex, \
        __FILE__, __LINE__, &gtest_dt)) { \
      goto GTEST_CONCAT_TOKEN_(gtest_label_, __LINE__); \
    } \
    if (gtest_dt != NULL) { \
      ::testing::internal::scoped_ptr< ::testing::internal::DeathTest> \
          gtest_dt_ptr(gtest_dt); \
      switch (gtest_dt->AssumeRole()) { \
        case ::testing::internal::DeathTest::OVERSEE_TEST: \
          if (!gtest_dt->Passed(predicate(gtest_dt->Wait()))) { \
            goto GTEST_CONCAT_TOKEN_(gtest_label_, __LINE__); \
          } \
          break; \
        case ::testing::internal::DeathTest::EXECUTE_TEST: { \
          ::testing::internal::DeathTest::ReturnSentinel \
              gtest_sentinel(gtest_dt); \
          GTEST_EXECUTE_DEATH_TEST_STATEMENT_(statement, gtest_dt); \
          gtest_dt->Abort(::testing::internal::DeathTest::TEST_DID_NOT_DIE); \
          break; \
        } \
        default: \
          break; \
      } \
    } \
  } else \
    GTEST_CONCAT_TOKEN_(gtest_label_, __LINE__): \
      fail(::testing::internal::DeathTest::LastMessage())
// The symbol "fail" here expands to something into which a message
// can be streamed.

// This macro is for implementing ASSERT/EXPECT_DEBUG_DEATH when compiled in
// NDEBUG mode. In this case we need the statements to be executed and the macro
// must accept a streamed message even though the message is never printed.
// The regex object is not evaluated, but it is used to prevent "unused"
// warnings and to avoid an expression that doesn't compile in debug mode.
#define GTEST_EXECUTE_STATEMENT_(statement, regex)             \
  GTEST_AMBIGUOUS_ELSE_BLOCKER_                                 \
  if (::testing::internal::AlwaysTrue()) {                      \
    GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement); \
  } else if (!::testing::internal::AlwaysTrue()) {              \
    const ::testing::internal::RE& gtest_regex = (regex);      \
    static_cast<void>(gtest_regex);                             \
  } else                                                        \
    ::testing::Message()

// A class representing the parsed contents of the
// --gtest_internal_run_death_test flag, as it existed when
// RUN_ALL_TESTS was called.
class InternalRunDeathTestFlag {
 public:
  InternalRunDeathTestFlag(const std::string& a_file,
                           int a_line,
                           int an_index,
                           int a_write_fd)
      : file_(a_file), line_(a_line), index_(an_index),
        write_fd_(a_write_fd) {}

  ~InternalRunDeathTestFlag() {
    if (write_fd_ >= 0)
      posix::Close(write_fd_);
  }

  const std::string& file() const { return file_; }
  int line() const { return line_; }
  int index() const { return index_; }
  int write_fd() const { return write_fd_; }

 private:
  std::string file_;
  int line_;
  int index_;
  int write_fd_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(InternalRunDeathTestFlag);
};

// Returns a newly created InternalRunDeathTestFlag object with fields
// initialized from the GTEST_FLAG(internal_run_death_test) flag if
// the flag is specified; otherwise returns NULL.
InternalRunDeathTestFlag* ParseInternalRunDeathTestFlag();

#endif  // GTEST_HAS_DEATH_TEST

}  // namespace internal
}  // namespace testing

#endif  // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_DEATH_TEST_INTERNAL_H_
```

9.35 Dokumentacja pliku packages/Microsoft.googletest.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.

```
// Copyright 2008, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// Google Test filepath utilities
//
// This header file declares classes and functions used internally by
// Google Test.  They are subject to change without notice.
//
// This file is #included in gtest/internal/gtest-internal.h.
// Do not include this header file separately!

// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_FILEPATH_H_
#define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_FILEPATH_H_

#include "gtest/internal/gtest-string.h"

GTEST_DISABLE_MSC_WARNINGS_PUSH_(4251 \
/* class A needs to have dll-interface to be used by clients of class B */)

namespace testing {
namespace internal {

// FilePath - a class for file and directory pathname manipulation which
// handles platform-specific conventions (like the pathname separator).
// Used for helper functions for naming files in a directory for xml output.
// Except for Set methods, all methods are const or static, which provides an
// "immutable value object" -- useful for peace of mind.
// A FilePath with a value ending in a path separator ("like/this/") represents
// a directory, otherwise it is assumed to represent a file. In either case,
// it may or may not represent an actual file or directory in the file system.
// Names are NOT checked for syntax correctness -- no checking for illegal
// characters, malformed paths, etc.

class GTEST_API_ FilePath {
 public:
  FilePath() : pathname_("") { }
  FilePath(const FilePath& rhs) : pathname_(rhs.pathname_) { }

  explicit FilePath(const std::string& pathname) : pathname_(pathname) {
    Normalize();
  }

  FilePath& operator=(const FilePath& rhs) {
    Set(rhs);
    return *this;
  }

  void Set(const FilePath& rhs) {
    pathname_ = rhs.pathname_;
  }

  const std::string& string() const { return pathname_; }
  const char* c_str() const { return pathname_.c_str(); }

  // Returns the current working directory, or "" if unsuccessful.
  static FilePath GetCurrentDir();

  // Given directory = "dir", base_name = "test", number = 0,
  // extension = "xml", returns "dir/test.xml". If number is greater
  // than zero (e.g., 12), returns "dir/test_12.xml".
  // On Windows platform, uses \ as the separator rather than /.
  static FilePath MakeFileName(const FilePath& directory,
                               const FilePath& base_name,
                               int number,
                               const char* extension);

  // Given directory = "dir", relative_path = "test.xml",
  // returns "dir/test.xml".
  // On Windows, uses \ as the separator rather than /.
  static FilePath ConcatPaths(const FilePath& directory,
                              const FilePath& relative_path);

  // Returns a pathname for a file that does not currently exist. The pathname
  // will be directory/base_name.extension or
  // directory/base_name_<number>.extension if directory/base_name.extension
  // already exists. The number will be incremented until a pathname is found
  // that does not already exist.
  // Examples: 'dir/foo_test.xml' or 'dir/foo_test_1.xml'.
  // There could be a race condition if two or more processes are calling this
  // function at the same time -- they could both pick the same filename.
  static FilePath GenerateUniqueFileName(const FilePath& directory,
                                         const FilePath& base_name,
                                         const char* extension);

  // Returns true iff the path is "".
  bool IsEmpty() const { return pathname_.empty(); }

  // If input name has a trailing separator character, removes it and returns
  // the name, otherwise return the name string unmodified.
  // On Windows platform, uses \ as the separator, other platforms use /.
  FilePath RemoveTrailingPathSeparator() const;

  // Returns a copy of the FilePath with the directory part removed.
  // Example: FilePath("path/to/file").RemoveDirectoryName() returns
  // FilePath("file"). If there is no directory part ("just_a_file"), it returns
  // the FilePath unmodified. If there is no file part ("just_a_dir/") it
  // returns an empty FilePath ("").
  // On Windows platform, '\' is the path separator, otherwise it is '/'.
  FilePath RemoveDirectoryName() const;

  // RemoveFileName returns the directory path with the filename removed.
  // Example: FilePath("path/to/file").RemoveFileName() returns "path/to/".
  // If the FilePath is "a_file" or "/a_file", RemoveFileName returns
  // FilePath("./") or, on Windows, FilePath(".\\"). If the filepath does
  // not have a file, like "just/a/dir/", it returns the FilePath unmodified.
  // On Windows platform, '\' is the path separator, otherwise it is '/'.
  FilePath RemoveFileName() const;

  // Returns a copy of the FilePath with the case-insensitive extension removed.
  // Example: FilePath("dir/file.exe").RemoveExtension("EXE") returns
  // FilePath("dir/file"). If a case-insensitive extension is not
  // found, returns a copy of the original FilePath.
  FilePath RemoveExtension(const char* extension) const;

  // Creates directories so that path exists. Returns true if successful or if
  // the directories already exist; returns false if unable to create
  // directories for any reason. Will also return false if the FilePath does
  // not represent a directory (that is, it doesn't end with a path separator).
  bool CreateDirectoriesRecursively() const;

  // Create the directory so that path exists. Returns true if successful or
  // if the directory already exists; returns false if unable to create the
  // directory for any reason, including if the parent directory does not
  // exist. Not named "CreateDirectory" because that's a macro on Windows.
  bool CreateFolder() const;

  // Returns true if FilePath describes something in the file-system,
  // either a file, directory, or whatever, and that something exists.
  bool FileOrDirectoryExists() const;

  // Returns true if pathname describes a directory in the file-system
  // that exists.
  bool DirectoryExists() const;

  // Returns true if FilePath ends with a path separator, which indicates that
  // it is intended to represent a directory. Returns false otherwise.
  // This does NOT check that a directory (or file) actually exists.
  bool IsDirectory() const;

  // Returns true if pathname describes a root directory. (Windows has one
  // root directory per disk drive.)
  bool IsRootDirectory() const;

  // Returns true if pathname describes an absolute path.
  bool IsAbsolutePath() const;

 private:
  // Replaces multiple consecutive separators with a single separator.
  // For example, "bar///foo" becomes "bar/foo". Does not eliminate other
  // redundancies that might be in a pathname involving "." or "..".
  //
  // A pathname with multiple consecutive separators may occur either through
  // user error or as a result of some scripts or APIs that generate a pathname
  // with a trailing separator. On other platforms the same API or script
  // may NOT generate a pathname with a trailing "/". Then elsewhere that
  // pathname may have another "/" and pathname components added to it,
  // without checking for the separator already being there.
  // The script language and operating system may allow paths like "foo//bar"
  // but some of the functions in FilePath will not handle that correctly. In
  // particular, RemoveTrailingPathSeparator() only removes one separator, and
  // it is called in CreateDirectoriesRecursively() assuming that it will change
  // a pathname from directory syntax (trailing separator) to filename syntax.
  //
  // On Windows this method also replaces the alternate path separator '/' with
  // the primary path separator '\\', so that for example "bar\\/\\foo" becomes
  // "bar\\foo".

  void Normalize();

  // Returns a pointer to the last occurence of a valid path separator in
  // the FilePath. On Windows, for example, both '/' and '\' are valid path
  // separators. Returns NULL if no path separator was found.
  const char* FindLastPathSeparator() const;

  std::string pathname_;
};  // class FilePath

}  // namespace internal
}  // namespace testing

GTEST_DISABLE_MSC_WARNINGS_POP_()  //  4251

#endif  // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_FILEPATH_H_
```

9.37 Dokumentacja pliku packages/Microsoft.googletest.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::TypeIdHelper< 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::TypeId
- 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 &gtest_msg, const Message &user_msg)
- GTEST_API_ std::vector< EditType > testing::internal::edit_distance::CalculateOptimalEdits (const std↩ ::vector< size_t > &left, const std::vector< size_t > &right)
- GTEST_API_ std::vector< EditType > testing::internal::edit_distance::CalculateOptimalEdits (const std↩ ::vector< std::string > &left, const std::vector< std::string > &right)
- 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)
- 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>
  TypeId testing::internal::GetTypeId ()
- GTEST_API_ TypeId testing::internal::GetTestTypeId ()
- 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)
- GTEST_API_ bool testing::internal::SkipPrefix (const char *prefix, const char **pstr)
- 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)
```

Wartość:

```
    (sizeof(::testing::internal::IsNullLiteralHelper(x)) == 1)
```

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<\
            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.

```
// Copyright 2005, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// The Google C++ Testing and Mocking Framework (Google Test)
//
// This header file declares functions and macros used internally by
// Google Test.  They are subject to change without notice.

// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_INTERNAL_H_
#define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_INTERNAL_H_

#include "gtest/internal/gtest-port.h"

#if GTEST_OS_LINUX
# include <stdlib.h>
# include <sys/types.h>
# include <sys/wait.h>
# include <unistd.h>
#endif  // GTEST_OS_LINUX

#if GTEST_HAS_EXCEPTIONS
# include <stdexcept>
#endif

#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"

// Due to C++ preprocessor weirdness, we need double indirection to
// concatenate two tokens when one of them is __LINE__.  Writing
//
//   foo ## __LINE__
//
// will result in the token foo__LINE__, instead of foo followed by
// the current line number.  For more details, see
// http://www.parashift.com/c++-faq-lite/misc-technical-issues.html#faq-39.6
#define GTEST_CONCAT_TOKEN_(foo, bar) GTEST_CONCAT_TOKEN_IMPL_(foo, bar)
#define GTEST_CONCAT_TOKEN_IMPL_(foo, bar) foo ## bar

// Stringifies its argument.
#define GTEST_STRINGIFY_(name) #name

class ProtocolMessage;
namespace proto2 { class Message; }

namespace testing {

// Forward declarations.

class AssertionResult;                 // Result of an assertion.
class Message;                         // Represents a failure message.
class Test;                            // Represents a test.
class TestInfo;                        // Information about a test.
class TestPartResult;                  // Result of a test part.
class UnitTest;                        // A collection of test cases.

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

namespace internal {

```

```
struct TraceInfo;                      // Information about a trace point.
class TestInfoImpl;                    // Opaque implementation of TestInfo
class UnitTestImpl;                    // Opaque implementation of UnitTest

// The text used in failure messages to indicate the start of the
// stack trace.
GTEST_API_ extern const char kStackTraceMarker[];

// Two overloaded helpers for checking at compile time whether an
// expression is a null pointer literal (i.e. NULL or any 0-valued
// compile-time integral constant).  Their return values have
// different sizes, so we can use sizeof() to test which version is
// picked by the compiler.  These helpers have no implementations, as
// we only need their signatures.
//
// Given IsNullLiteralHelper(x), the compiler will pick the first
// version if x can be implicitly converted to Secret*, and pick the
// second version otherwise.  Since Secret is a secret and incomplete
// type, the only expression a user can write that has type Secret* is
// a null pointer literal.  Therefore, we know that x is a null
// pointer literal if and only if the first version is picked by the
// compiler.
char IsNullLiteralHelper(Secret* p);
char (&IsNullLiteralHelper(...))[2];  // NOLINT

// A compile-time bool constant that is true if and only if x is a
// null pointer literal (i.e. NULL or any 0-valued compile-time
// integral constant).
#ifdef GTEST_ELLIPSIS_NEEDS_POD_
// We lose support for NULL detection where the compiler doesn't like
// passing non-POD classes through ellipsis (...).
# define GTEST_IS_NULL_LITERAL_(x) false
#else
# define GTEST_IS_NULL_LITERAL_(x) \
    (sizeof(::testing::internal::IsNullLiteralHelper(x)) == 1)
#endif  // GTEST_ELLIPSIS_NEEDS_POD_

// Appends the user-supplied message to the Google-Test-generated message.
GTEST_API_ std::string AppendUserMessage(
    const std::string& gtest_msg, const Message& user_msg);

#if GTEST_HAS_EXCEPTIONS

GTEST_DISABLE_MSC_WARNINGS_PUSH_(4275 \
/* an exported class was derived from a class that was not exported */)

// This exception is thrown by (and only by) a failed Google Test
// assertion when GTEST_FLAG(throw_on_failure) is true (if exceptions
// are enabled).  We derive it from std::runtime_error, which is for
// errors presumably detectable only at run time.  Since
// std::runtime_error inherits from std::exception, many testing
// frameworks know how to extract and print the message inside it.
class GTEST_API_ GoogleTestFailureException : public ::std::runtime_error {
 public:
  explicit GoogleTestFailureException(const TestPartResult& failure);
};

GTEST_DISABLE_MSC_WARNINGS_POP_()  //  4275

#endif  // GTEST_HAS_EXCEPTIONS

namespace edit_distance {
// Returns the optimal edits to go from 'left' to 'right'.
// All edits cost the same, with replace having lower priority than
// add/remove.
// Simple implementation of the Wagner-Fischer algorithm.
// See http://en.wikipedia.org/wiki/Wagner-Fischer_algorithm
enum EditType { kMatch, kAdd, kRemove, kReplace };
GTEST_API_ std::vector<EditType> CalculateOptimalEdits(
    const std::vector<size_t>& left, const std::vector<size_t>& right);

// Same as above, but the input is represented as strings.
GTEST_API_ std::vector<EditType> CalculateOptimalEdits(
    const std::vector<std::string>& left,
    const std::vector<std::string>& right);

// Create a diff of the input strings in Unified diff format.
GTEST_API_ std::string CreateUnifiedDiff(const std::vector<std::string>& left,
                                         const std::vector<std::string>& right,
                                         size_t context = 2);

}  // namespace edit_distance

// Calculate the diff between 'left' and 'right' and return it in unified diff
// format.
// If not null, stores in 'total_line_count' the total number of lines found
// in left + right.
```

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

// Constructs and returns the message for an equality assertion
// (e.g. ASSERT_EQ, EXPECT_STREQ, etc) failure.
//
// The first four parameters are the expressions used in the assertion
// and their values, as strings.  For example, for ASSERT_EQ(foo, bar)
// where foo is 5 and bar is 6, we have:
//
//   expected_expression: "foo"
//   actual_expression:   "bar"
//   expected_value:      "5"
//   actual_value:        "6"
//
// The ignoring_case parameter is true iff the assertion is a
// *_STRCASEEQ*.  When it's true, the string " (ignoring case)" will
// be inserted into the message.
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);

// Constructs a failure message for Boolean assertions such as EXPECT_TRUE.
GTEST_API_ std::string GetBoolAssertionFailureMessage(
    const AssertionResult& assertion_result,
    const char* expression_text,
    const char* actual_predicate_value,
    const char* expected_predicate_value);

// This template class represents an IEEE floating-point number
// (either single-precision or double-precision, depending on the
// template parameters).
//
// The purpose of this class is to do more sophisticated number
// comparison.  (Due to round-off error, etc, it's very unlikely that
// two floating-points will be equal exactly.  Hence a naive
// comparison by the == operation often doesn't work.)
//
// Format of IEEE floating-point:
//
//   The most-significant bit being the leftmost, an IEEE
//   floating-point looks like
//
//     sign_bit exponent_bits fraction_bits
//
//   Here, sign_bit is a single bit that designates the sign of the
//   number.
//
//   For float, there are 8 exponent bits and 23 fraction bits.
//
//   For double, there are 11 exponent bits and 52 fraction bits.
//
//   More details can be found at
//   http://en.wikipedia.org/wiki/IEEE_floating-point_standard.
//
// Template parameter:
//
//   RawType: the raw floating-point type (either float or double)
template <typename RawType>
class FloatingPoint {
 public:
  // Defines the unsigned integer type that has the same size as the
  // floating point number.
  typedef typename TypeWithSize<sizeof(RawType)>::UInt Bits;

  // Constants.

  // # of bits in a number.
  static const size_t kBitCount = 8*sizeof(RawType);

  // # of fraction bits in a number.
  static const size_t kFractionBitCount =
    std::numeric_limits<RawType>::digits - 1;

  // # of exponent bits in a number.
  static const size_t kExponentBitCount = kBitCount - 1 - kFractionBitCount;

  // The mask for the sign bit.
  static const Bits kSignBitMask = static_cast<Bits>(1) « (kBitCount - 1);

  // The mask for the fraction bits.
  static const Bits kFractionBitMask =
    ~static_cast<Bits>(0) » (kExponentBitCount + 1);

  // The mask for the exponent bits.
  static const Bits kExponentBitMask = ~(kSignBitMask | kFractionBitMask);

  // How many ULP's (Units in the Last Place) we want to tolerate when
  // comparing two numbers.  The larger the value, the more error we
  // allow.  A 0 value means that two numbers must be exactly the same
  // to be considered equal.
  //
  // The maximum error of a single floating-point operation is 0.5
  // units in the last place.  On Intel CPU's, all floating-point
  // calculations are done with 80-bit precision, while double has 64
  // bits.  Therefore, 4 should be enough for ordinary use.
  //
  // See the following article for more details on ULP:
  // http://randomascii.wordpress.com/2012/02/25/comparing-floating-point-numbers-2012-edition/
  static const size_t kMaxUlps = 4;

  // Constructs a FloatingPoint from a raw floating-point number.
  //
  // On an Intel CPU, passing a non-normalized NAN (Not a Number)
  // around may change its bits, although the new value is guaranteed
  // to be also a NAN.  Therefore, don't expect this constructor to
  // preserve the bits in x when x is a NAN.
  explicit FloatingPoint(const RawType& x) { u_.value_ = x; }

  // Static methods

  // Reinterprets a bit pattern as a floating-point number.
  //
  // This function is needed to test the AlmostEquals() method.
  static RawType ReinterpretBits(const Bits bits) {
    FloatingPoint fp(0);
    fp.u_.bits_ = bits;
    return fp.u_.value_;
  }

  // Returns the floating-point number that represent positive infinity.
  static RawType Infinity() {
    return ReinterpretBits(kExponentBitMask);
  }

  // Returns the maximum representable finite floating-point number.
  static RawType Max();

  // Non-static methods

  // Returns the bits that represents this number.
  const Bits &bits() const { return u_.bits_; }

  // Returns the exponent bits of this number.
  Bits exponent_bits() const { return kExponentBitMask & u_.bits_; }

  // Returns the fraction bits of this number.
  Bits fraction_bits() const { return kFractionBitMask & u_.bits_; }

  // Returns the sign bit of this number.
  Bits sign_bit() const { return kSignBitMask & u_.bits_; }

  // Returns true iff this is NAN (not a number).
  bool is_nan() const {
    // It's a NAN if the exponent bits are all ones and the fraction
    // bits are not entirely zeros.
    return (exponent_bits() == kExponentBitMask) && (fraction_bits() != 0);
  }

  // Returns true iff this number is at most kMaxUlps ULP's away from
  // rhs.  In particular, this function:
  //
  //   - returns false if either number is (or both are) NAN.
  //   - treats really large numbers as almost equal to infinity.
  //   - thinks +0.0 and -0.0 are 0 DLP's apart.
  bool AlmostEquals(const FloatingPoint& rhs) const {
    // The IEEE standard says that any comparison operation involving
    // a NAN must return false.
    if (is_nan() || rhs.is_nan()) return false;

    return DistanceBetweenSignAndMagnitudeNumbers(u_.bits_, rhs.u_.bits_)
        <= kMaxUlps;
  }

 private:
  // The data type used to store the actual floating-point number.
  union FloatingPointUnion {
    RawType value_;  // The raw floating-point number.
    Bits bits_;      // The bits that represent the number.
  };

  // Converts an integer from the sign-and-magnitude representation to
  // the biased representation.  More precisely, let N be 2 to the
  // power of (kBitCount - 1), an integer x is represented by the
  // unsigned number x + N.
  //
  // For instance,
  //
  //   -N + 1 (the most negative number representable using
  //          sign-and-magnitude) is represented by 1;
  //   0      is represented by N; and
  //   N - 1  (the biggest number representable using
  //          sign-and-magnitude) is represented by 2N - 1.
  //
  // Read http://en.wikipedia.org/wiki/Signed_number_representations
  // for more details on signed number representations.
  static Bits SignAndMagnitudeToBiased(const Bits &sam) {
    if (kSignBitMask & sam) {
      // sam represents a negative number.
      return ~sam + 1;
    } else {
      // sam represents a positive number.
      return kSignBitMask | sam;
    }
  }

  // Given two numbers in the sign-and-magnitude representation,
  // returns the distance between them as an unsigned number.
  static Bits DistanceBetweenSignAndMagnitudeNumbers(const Bits &sam1,
                                                     const Bits &sam2) {
    const Bits biased1 = SignAndMagnitudeToBiased(sam1);
    const Bits biased2 = SignAndMagnitudeToBiased(sam2);
    return (biased1 >= biased2) ? (biased1 - biased2) : (biased2 - biased1);
  }

  FloatingPointUnion u_;
};

// We cannot use std::numeric_limits<T>::max() as it clashes with the max()
// macro defined by <windows.h>.
template <>
inline float FloatingPoint<float>::Max() { return FLT_MAX; }
template <>
inline double FloatingPoint<double>::Max() { return DBL_MAX; }

// Typedefs the instances of the FloatingPoint template class that we
// care to use.
typedef FloatingPoint<float> Float;
typedef FloatingPoint<double> Double;

// In order to catch the mistake of putting tests that use different
// test fixture classes in the same test case, we need to assign
// unique IDs to fixture classes and compare them.  The TypeId type is
// used to hold such IDs.  The user should treat TypeId as an opaque
// type: the only operation allowed on TypeId values is to compare
// them for equality using the == operator.
typedef const void* TypeId;

template <typename T>
class TypeIdHelper {
 public:
  // dummy_ must not have a const type.  Otherwise an overly eager
  // compiler (e.g. MSVC 7.1 & 8.0) may try to merge
  // TypeIdHelper<T>::dummy_ for different Ts as an "optimization".
  static bool dummy_;
};

template <typename T>
bool TypeIdHelper<T>::dummy_ = false;

// GetTypeId<T>() returns the ID of type T.  Different values will be
// returned for different types.  Calling the function twice with the
// same type argument is guaranteed to return the same ID.
template <typename T>
TypeId GetTypeId() {
  // The compiler is required to allocate a different
  // TypeIdHelper<T>::dummy_ variable for each T used to instantiate
  // the template.  Therefore, the address of dummy_ is guaranteed to
  // be unique.
  return &(TypeIdHelper<T>::dummy_);
}

// Returns the type ID of ::testing::Test.  Always call this instead
// of GetTypeId< ::testing::Test>() to get the type ID of
// ::testing::Test, as the latter may give the wrong result due to a
// suspected linker bug when compiling Google Test as a Mac OS X
// framework.
GTEST_API_ TypeId GetTestTypeId();
```

```cpp

// Defines the abstract factory interface that creates instances
// of a Test object.
class TestFactoryBase {
 public:
  virtual ~TestFactoryBase() {}

  // Creates a test instance to run. The instance is both created and destroyed
  // within TestInfoImpl::Run()
  virtual Test* CreateTest() = 0;

 protected:
  TestFactoryBase() {}

 private:
  GTEST_DISALLOW_COPY_AND_ASSIGN_(TestFactoryBase);
};

// This class provides implementation of TeastFactoryBase interface.
// It is used in TEST and TEST_F macros.
template <class TestClass>
class TestFactoryImpl : public TestFactoryBase {
 public:
  virtual Test* CreateTest() { return new TestClass; }
};

#if GTEST_OS_WINDOWS

// Predicate-formatters for implementing the HRESULT checking macros
// {ASSERT|EXPECT}_HRESULT_{SUCCEEDED|FAILED}
// We pass a long instead of HRESULT to avoid causing an
// include dependency for the HRESULT type.
GTEST_API_ AssertionResult IsHRESULTSuccess(const char* expr,
                                            long hr);  // NOLINT
GTEST_API_ AssertionResult IsHRESULTFailure(const char* expr,
                                            long hr);  // NOLINT

#endif  // GTEST_OS_WINDOWS

// Types of SetUpTestCase() and TearDownTestCase() functions.
typedef void (*SetUpTestCaseFunc)();
typedef void (*TearDownTestCaseFunc)();

struct CodeLocation {
  CodeLocation(const std::string& a_file, int a_line)
      : file(a_file), line(a_line) {}

  std::string file;
  int line;
};

// Creates a new TestInfo object and registers it with Google Test;
// returns the created object.
//
// Arguments:
//
//   test_case_name:   name of the test case
//   name:             name of the test
//   type_param        the name of the test's type parameter, or NULL if
//                     this is not a typed or a type-parameterized test.
//   value_param       text representation of the test's value parameter,
//                     or NULL if this is not a type-parameterized test.
//   code_location:    code location where the test is defined
//   fixture_class_id: ID of the test fixture class
//   set_up_tc:        pointer to the function that sets up the test case
//   tear_down_tc:     pointer to the function that tears down the test case
//   factory:          pointer to the factory that creates a test object.
//                     The newly created TestInfo instance will assume
//                     ownership of the factory object.
GTEST_API_ TestInfo* 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);

// If *pstr starts with the given prefix, modifies *pstr to be right
// past the prefix and returns true; otherwise leaves *pstr unchanged
// and returns false.  None of pstr, *pstr, and prefix can be NULL.
GTEST_API_ bool SkipPrefix(const char* prefix, const char** pstr);

#if GTEST_HAS_TYPED_TEST || GTEST_HAS_TYPED_TEST_P

```

```
GTEST_DISABLE_MSC_WARNINGS_PUSH_(4251 \
/* class A needs to have dll-interface to be used by clients of class B */)

// State of the definition of a type-parameterized test case.
class GTEST_API_ TypedTestCasePState {
 public:
  TypedTestCasePState() : registered_(false) {}

  // Adds the given test name to defined_test_names_ and return true
  // if the test case hasn't been registered; otherwise aborts the
  // program.
  bool AddTestName(const char* file, int line, const char* case_name,
                   const char* test_name) {
    if (registered_) {
      fprintf(stderr, "%s Test %s must be defined before "
              "REGISTER_TYPED_TEST_CASE_P(%s, ...).\n",
              FormatFileLocation(file, line).c_str(), test_name, case_name);
      fflush(stderr);
      posix::Abort();
    }
    registered_tests_.insert(
        ::std::make_pair(test_name, CodeLocation(file, line)));
    return true;
  }

  bool TestExists(const std::string& test_name) const {
    return registered_tests_.count(test_name) > 0;
  }

  const CodeLocation& GetCodeLocation(const std::string& test_name) const {
    RegisteredTestsMap::const_iterator it = registered_tests_.find(test_name);
    GTEST_CHECK_(it != registered_tests_.end());
    return it->second;
  }

  // Verifies that registered_tests match the test names in
  // defined_test_names_; returns registered_tests if successful, or
  // aborts the program otherwise.
  const char* VerifyRegisteredTestNames(
      const char* file, int line, const char* registered_tests);

 private:
  typedef ::std::map<std::string, CodeLocation> RegisteredTestsMap;

  bool registered_;
  RegisteredTestsMap registered_tests_;
};

GTEST_DISABLE_MSC_WARNINGS_POP_()  //  4251

// Skips to the first non-space char after the first comma in 'str';
// returns NULL if no comma is found in 'str'.
inline const char* SkipComma(const char* str) {
  const char* comma = strchr(str, ',');
  if (comma == NULL) {
    return NULL;
  }
  while (IsSpace(*(++comma))) {}
  return comma;
}

// Returns the prefix of 'str' before the first comma in it; returns
// the entire string if it contains no comma.
inline std::string GetPrefixUntilComma(const char* str) {
  const char* comma = strchr(str, ',');
  return comma == NULL ? str : std::string(str, comma);
}

// Splits a given string on a given delimiter, populating a given
// vector with the fields.
void SplitString(const ::std::string& str, char delimiter,
                 ::std::vector< ::std::string>* dest);

// The default argument to the template below for the case when the user does
// not provide a name generator.
struct DefaultNameGenerator {
  template <typename T>
  static std::string GetName(int i) {
    return StreamableToString(i);
  }
};

template <typename Provided = DefaultNameGenerator>
struct NameGeneratorSelector {
  typedef Provided type;
};

```

```
template <typename NameGenerator>
void GenerateNamesRecursively(Types0, std::vector<std::string>*, int) {}

template <typename NameGenerator, typename Types>
void GenerateNamesRecursively(Types, std::vector<std::string>* result, int i) {
  result->push_back(NameGenerator::template GetName<typename Types::Head>(i));
  GenerateNamesRecursively<NameGenerator>(typename Types::Tail(), result,
                                          i + 1);
}

template <typename NameGenerator, typename Types>
std::vector<std::string> GenerateNames() {
  std::vector<std::string> result;
  GenerateNamesRecursively<NameGenerator>(Types(), &result, 0);
  return result;
}

// TypeParameterizedTest<Fixture, TestSel, Types>::Register()
// registers a list of type-parameterized tests with Google Test.  The
// return value is insignificant - we just need to return something
// such that we can call this function in a namespace scope.
//
// Implementation note: The GTEST_TEMPLATE_ macro declares a template
// template parameter.  It's defined in gtest-type-util.h.
template <GTEST_TEMPLATE_ Fixture, class TestSel, typename Types>
class TypeParameterizedTest {
 public:
  // 'index' is the index of the test in the type list 'Types'
  // specified in INSTANTIATE_TYPED_TEST_CASE_P(Prefix, TestCase,
  // Types).  Valid values for 'index' are [0, N - 1] where N is the
  // length of Types.
  static bool Register(const char* prefix, const CodeLocation& code_location,
                       const char* case_name, const char* test_names, int index,
                       const std::vector<std::string>& type_names =
                           GenerateNames<DefaultNameGenerator, Types>()) {
    typedef typename Types::Head Type;
    typedef Fixture<Type> FixtureClass;
    typedef typename GTEST_BIND_(TestSel, Type) TestClass;

    // First, registers the first type-parameterized test in the type
    // list.
    MakeAndRegisterTestInfo(
        (std::string(prefix) + (prefix[0] == '\0' ? "" : "/") + case_name +
         "/" + type_names[index])
            .c_str(),
        StripTrailingSpaces(GetPrefixUntilComma(test_names)).c_str(),
        GetTypeName<Type>().c_str(),
        NULL,  // No value parameter.
        code_location, GetTypeId<FixtureClass>(), TestClass::SetUpTestCase,
        TestClass::TearDownTestCase, new TestFactoryImpl<TestClass>);

    // Next, recurses (at compile time) with the tail of the type list.
    return TypeParameterizedTest<Fixture, TestSel,
                                 typename Types::Tail>::Register(prefix,
                                                                 code_location,
                                                                 case_name,
                                                                 test_names,
                                                                 index + 1,
                                                                 type_names);
  }
};

// The base case for the compile time recursion.
template <GTEST_TEMPLATE_ Fixture, class TestSel>
class TypeParameterizedTest<Fixture, TestSel, Types0> {
 public:
  static bool Register(const char* /*prefix*/, const CodeLocation&,
                       const char* /*case_name*/, const char* /*test_names*/,
                       int /*index*/,
                       const std::vector<std::string>& =
                           std::vector<std::string>() /*type_names*/) {
    return true;
  }
};

// TypeParameterizedTestCase<Fixture, Tests, Types>::Register()
// registers *all combinations* of 'Tests' and 'Types' with Google
// Test.  The return value is insignificant - we just need to return
// something such that we can call this function in a namespace scope.
template <GTEST_TEMPLATE_ Fixture, typename Tests, typename Types>
class TypeParameterizedTestCase {
 public:
  static bool Register(const char* prefix, CodeLocation code_location,
                       const TypedTestCasePState* state, const char* case_name,
                       const char* test_names,
                       const std::vector<std::string>& type_names =
                           GenerateNames<DefaultNameGenerator, Types>()) {
```

```
    std::string test_name = StripTrailingSpaces(
        GetPrefixUntilComma(test_names));
    if (!state->TestExists(test_name)) {
      fprintf(stderr, "Failed to get code location for test %s.%s at %s.",
              case_name, test_name.c_str(),
              FormatFileLocation(code_location.file.c_str(),
                                 code_location.line).c_str());
      fflush(stderr);
      posix::Abort();
    }
    const CodeLocation& test_location = state->GetCodeLocation(test_name);

    typedef typename Tests::Head Head;

    // First, register the first test in 'Test' for each type in 'Types'.
    TypeParameterizedTest<Fixture, Head, Types>::Register(
        prefix, test_location, case_name, test_names, 0, type_names);

    // Next, recurses (at compile time) with the tail of the test list.
    return TypeParameterizedTestCase<Fixture, typename Tests::Tail,
                                     Types>::Register(prefix, code_location,
                                                      state, case_name,
                                                      SkipComma(test_names),
                                                      type_names);
  }
};

// The base case for the compile time recursion.
template <GTEST_TEMPLATE_ Fixture, typename Types>
class TypeParameterizedTestCase<Fixture, Templates0, Types> {
 public:
  static bool Register(const char* /*prefix*/, const CodeLocation&,
                       const TypedTestCasePState* /*state*/,
                       const char* /*case_name*/, const char* /*test_names*/,
                       const std::vector<std::string>& =
                           std::vector<std::string>() /*type_names*/) {
    return true;
  }
};

#endif  // GTEST_HAS_TYPED_TEST || GTEST_HAS_TYPED_TEST_P

// Returns the current OS stack trace as an std::string.
//
// The maximum number of stack frames to be included is specified by
// the gtest_stack_trace_depth flag.  The skip_count parameter
// specifies the number of top frames to be skipped, which doesn't
// count against the number of frames to be included.
//
// For example, if Foo() calls Bar(), which in turn calls
// GetCurrentOsStackTraceExceptTop(..., 1), Foo() will be included in
// the trace but Bar() and GetCurrentOsStackTraceExceptTop() won't.
GTEST_API_ std::string GetCurrentOsStackTraceExceptTop(
    UnitTest* unit_test, int skip_count);

// Helpers for suppressing warnings on unreachable code or constant
// condition.

// Always returns true.
GTEST_API_ bool AlwaysTrue();

// Always returns false.
inline bool AlwaysFalse() { return !AlwaysTrue(); }

// Helper for suppressing false warning from Clang on a const char*
// variable declared in a conditional expression always being NULL in
// the else branch.
struct GTEST_API_ ConstCharPtr {
  ConstCharPtr(const char* str) : value(str) {}
  operator bool() const { return true; }
  const char* value;
};

// A simple Linear Congruential Generator for generating random
// numbers with a uniform distribution.  Unlike rand() and srand(), it
// doesn't use global state (and therefore can't interfere with user
// code).  Unlike rand_r(), it's portable.  An LCG isn't very random,
// but it's good enough for our purposes.
class GTEST_API_ Random {
 public:
  static const UInt32 kMaxRange = 1u « 31;

  explicit Random(UInt32 seed) : state_(seed) {}

  void Reseed(UInt32 seed) { state_ = seed; }

  // Generates a random number from [0, range).  Crashes if 'range' is
  // 0 or greater than kMaxRange.
  UInt32 Generate(UInt32 range);

 private:
  UInt32 state_;
  GTEST_DISALLOW_COPY_AND_ASSIGN_(Random);
};

// Defining a variable of type CompileAssertTypesEqual<T1, T2> will cause a
// compiler error iff T1 and T2 are different types.
template <typename T1, typename T2>
struct CompileAssertTypesEqual;

template <typename T>
struct CompileAssertTypesEqual<T, T> {
};

// Removes the reference from a type if it is a reference type,
// otherwise leaves it unchanged.  This is the same as
// tr1::remove_reference, which is not widely available yet.
template <typename T>
struct RemoveReference { typedef T type; };  // NOLINT
template <typename T>
struct RemoveReference<T&> { typedef T type; };  // NOLINT

// A handy wrapper around RemoveReference that works when the argument
// T depends on template parameters.
#define GTEST_REMOVE_REFERENCE_(T) \
    typename ::testing::internal::RemoveReference<T>::type

// Removes const from a type if it is a const type, otherwise leaves
// it unchanged.  This is the same as tr1::remove_const, which is not
// widely available yet.
template <typename T>
struct RemoveConst { typedef T type; };  // NOLINT
template <typename T>
struct RemoveConst<const T> { typedef T type; };  // NOLINT

// MSVC 8.0, Sun C++, and IBM XL C++ have a bug which causes the above
// definition to fail to remove the const in 'const int[3]' and 'const
// char[3][4]'.  The following specialization works around the bug.
template <typename T, size_t N>
struct RemoveConst<const T[N]> {
  typedef typename RemoveConst<T>::type type[N];
};

#if defined(_MSC_VER) && _MSC_VER < 1400
// This is the only specialization that allows VC++ 7.1 to remove const in
// 'const int[3] and 'const int[3][4]'.  However, it causes trouble with GCC
// and thus needs to be conditionally compiled.
template <typename T, size_t N>
struct RemoveConst<T[N]> {
  typedef typename RemoveConst<T>::type type[N];
};
#endif

// A handy wrapper around RemoveConst that works when the argument
// T depends on template parameters.
#define GTEST_REMOVE_CONST_(T) \
    typename ::testing::internal::RemoveConst<T>::type

// Turns const U&, U&, const U, and U all into U.
#define GTEST_REMOVE_REFERENCE_AND_CONST_(T) \
    GTEST_REMOVE_CONST_(GTEST_REMOVE_REFERENCE_(T))

// ImplicitlyConvertible<From, To>::value is a compile-time bool
// constant that's true iff type From can be implicitly converted to
// type To.
template <typename From, typename To>
class ImplicitlyConvertible {
 private:
  // We need the following helper functions only for their types.
  // They have no implementations.

  // MakeFrom() is an expression whose type is From.  We cannot simply
  // use From(), as the type From may not have a public default
  // constructor.
  static typename AddReference<From>::type MakeFrom();

  // These two functions are overloaded.  Given an expression
  // Helper(x), the compiler will pick the first version if x can be
  // implicitly converted to type To; otherwise it will pick the
  // second version.
  //
  // The first version returns a value of size 1, and the second
  // version returns a value of size 2.  Therefore, by checking the
  // size of Helper(x), which can be done at compile time, we can tell
  // which version of Helper() is used, and hence whether x can be
  // implicitly converted to type To.
  static char Helper(To);
  static char (&Helper(...))[2];  // NOLINT

  // We have to put the 'public' section after the 'private' section,
  // or MSVC refuses to compile the code.
 public:
#if defined(__BORLANDC__)
  // C++Builder cannot use member overload resolution during template
  // instantiation.  The simplest workaround is to use its C++0x type traits
  // functions (C++Builder 2009 and above only).
  static const bool value = __is_convertible(From, To);
#else
  // MSVC warns about implicitly converting from double to int for
  // possible loss of data, so we need to temporarily disable the
  // warning.
  GTEST_DISABLE_MSC_WARNINGS_PUSH_(4244)
  static const bool value =
      sizeof(Helper(ImplicitlyConvertible::MakeFrom())) == 1;
  GTEST_DISABLE_MSC_WARNINGS_POP_()
#endif  // __BORLANDC__
};
template <typename From, typename To>
const bool ImplicitlyConvertible<From, To>::value;

// IsAProtocolMessage<T>::value is a compile-time bool constant that's
// true iff T is type ProtocolMessage, proto2::Message, or a subclass
// of those.
template <typename T>
struct IsAProtocolMessage
    : public bool_constant<
  ImplicitlyConvertible<const T*, const ::ProtocolMessage*>::value ||
  ImplicitlyConvertible<const T*, const ::proto2::Message*>::value> {
};

// When the compiler sees expression IsContainerTest<C>(0), if C is an
// STL-style container class, the first overload of IsContainerTest
// will be viable (since both C::iterator* and C::const_iterator* are
// valid types and NULL can be implicitly converted to them).  It will
// be picked over the second overload as 'int' is a perfect match for
// the type of argument 0.  If C::iterator or C::const_iterator is not
// a valid type, the first overload is not viable, and the second
// overload will be picked.  Therefore, we can determine whether C is
// a container class by checking the type of IsContainerTest<C>(0).
// The value of the expression is insignificant.
//
// In C++11 mode we check the existence of a const_iterator and that an
// iterator is properly implemented for the container.
//
// For pre-C++11 that we look for both C::iterator and C::const_iterator.
// The reason is that C++ injects the name of a class as a member of the
// class itself (e.g. you can refer to class iterator as either
// 'iterator' or 'iterator::iterator').  If we look for C::iterator
// only, for example, we would mistakenly think that a class named
// iterator is an STL container.
//
// Also note that the simpler approach of overloading
// IsContainerTest(typename C::const_iterator*) and
// IsContainerTest(...) doesn't work with Visual Age C++ and Sun C++.
typedef int IsContainer;
#if GTEST_LANG_CXX11
template <class C,
          class Iterator = decltype(::std::declval<const C&>().begin()),
          class = decltype(::std::declval<const C&>().end()),
          class = decltype(++::std::declval<Iterator&>()),
          class = decltype(*::std::declval<Iterator>()),
          class = typename C::const_iterator>
IsContainer IsContainerTest(int /* dummy */) {
  return 0;
}
#else
template <class C>
IsContainer IsContainerTest(int /* dummy */,
                            typename C::iterator* /* it */ = NULL,
                            typename C::const_iterator* /* const_it */ = NULL) {
  return 0;
}
#endif  // GTEST_LANG_CXX11

typedef char IsNotContainer;
template <class C>
IsNotContainer IsContainerTest(long /* dummy */) { return '\0'; }

// Trait to detect whether a type T is a hash table.
// The heuristic used is that the type contains an inner type `hasher` and does
// not contain an inner type `reverse_iterator`.
// If the container is iterable in reverse, then order might actually matter.
template <typename T>
struct IsHashTable {
 private:
  template <typename U>
  static char test(typename U::hasher*, typename U::reverse_iterator*);
  template <typename U>
  static int test(typename U::hasher*, ...);
  template <typename U>
  static char test(...);

 public:
  static const bool value = sizeof(test<T>(0, 0)) == sizeof(int);
};

template <typename T>
const bool IsHashTable<T>::value;

template<typename T>
struct VoidT {
    typedef void value_type;
};

template <typename T, typename = void>
struct HasValueType : false_type {};
template <typename T>
struct HasValueType<T, VoidT<typename T::value_type> > : true_type {
};

template <typename C,
          bool = sizeof(IsContainerTest<C>(0)) == sizeof(IsContainer),
          bool = HasValueType<C>::value>
struct IsRecursiveContainerImpl;

template <typename C, bool HV>
struct IsRecursiveContainerImpl<C, false, HV> : public false_type {};

// Since the IsRecursiveContainerImpl depends on the IsContainerTest we need to
// obey the same inconsistencies as the IsContainerTest, namely check if
// something is a container is relying on only const_iterator in C++11 and
// is relying on both const_iterator and iterator otherwise
template <typename C>
struct IsRecursiveContainerImpl<C, true, false> : public false_type {};

template <typename C>
struct IsRecursiveContainerImpl<C, true, true> {
  #if GTEST_LANG_CXX11
  typedef typename IteratorTraits<typename C::const_iterator>::value_type
      value_type;
#else
  typedef typename IteratorTraits<typename C::iterator>::value_type value_type;
#endif
  typedef is_same<value_type, C> type;
};

// IsRecursiveContainer<Type> is a unary compile-time predicate that
// evaluates whether C is a recursive container type. A recursive container
// type is a container type whose value_type is equal to the container type
// itself. An example for a recursive container type is
// boost::filesystem::path, whose iterator has a value_type that is equal to
// boost::filesystem::path.
template <typename C>
struct IsRecursiveContainer : public IsRecursiveContainerImpl<C>::type {};

// EnableIf<condition>::type is void when 'Cond' is true, and
// undefined when 'Cond' is false.  To use SFINAE to make a function
// overload only apply when a particular expression is true, add
// "typename EnableIf<expression>::type* = 0" as the last parameter.
template<bool> struct EnableIf;
template<> struct EnableIf<true> { typedef void type; };  // NOLINT

// Utilities for native arrays.

// ArrayEq() compares two k-dimensional native arrays using the
// elements' operator==, where k can be any integer >= 0.  When k is
// 0, ArrayEq() degenerates into comparing a single pair of values.

template <typename T, typename U>
bool ArrayEq(const T* lhs, size_t size, const U* rhs);

// This generic version is used when k is 0.
template <typename T, typename U>
inline bool ArrayEq(const T& lhs, const U& rhs) { return lhs == rhs; }

// This overload is used when k >= 1.
template <typename T, typename U, size_t N>
inline bool ArrayEq(const T(&lhs)[N], const U(&rhs)[N]) {
  return internal::ArrayEq(lhs, N, rhs);
}

// This helper reduces code bloat.  If we instead put its logic inside
// the previous ArrayEq() function, arrays with different sizes would
// lead to different copies of the template code.
template <typename T, typename U>
bool ArrayEq(const T* lhs, size_t size, const U* rhs) {
  for (size_t i = 0; i != size; i++) {
    if (!internal::ArrayEq(lhs[i], rhs[i]))
      return false;
  }
  return true;
}

// Finds the first element in the iterator range [begin, end) that
// equals elem.  Element may be a native array type itself.
template <typename Iter, typename Element>
Iter ArrayAwareFind(Iter begin, Iter end, const Element& elem) {
  for (Iter it = begin; it != end; ++it) {
    if (internal::ArrayEq(*it, elem))
      return it;
  }
  return end;
}

// CopyArray() copies a k-dimensional native array using the elements'
// operator=, where k can be any integer >= 0.  When k is 0,
// CopyArray() degenerates into copying a single value.

template <typename T, typename U>
void CopyArray(const T* from, size_t size, U* to);

// This generic version is used when k is 0.
template <typename T, typename U>
inline void CopyArray(const T& from, U* to) { *to = from; }

// This overload is used when k >= 1.
template <typename T, typename U, size_t N>
inline void CopyArray(const T(&from)[N], U(*to)[N]) {
  internal::CopyArray(from, N, *to);
}

// This helper reduces code bloat.  If we instead put its logic inside
// the previous CopyArray() function, arrays with different sizes
// would lead to different copies of the template code.
template <typename T, typename U>
void CopyArray(const T* from, size_t size, U* to) {
  for (size_t i = 0; i != size; i++) {
    internal::CopyArray(from[i], to + i);
  }
}

// The relation between an NativeArray object (see below) and the
// native array it represents.
// We use 2 different structs to allow non-copyable types to be used, as long
// as RelationToSourceReference() is passed.
struct RelationToSourceReference {};
struct RelationToSourceCopy {};

// Adapts a native array to a read-only STL-style container.  Instead
// of the complete STL container concept, this adaptor only implements
// members useful for Google Mock's container matchers.  New members
// should be added as needed.  To simplify the implementation, we only
// support Element being a raw type (i.e. having no top-level const or
// reference modifier).  It's the client's responsibility to satisfy
// this requirement.  Element can be an array type itself (hence
// multi-dimensional arrays are supported).
template <typename Element>
class NativeArray {
 public:
  // STL-style container typedefs.
  typedef Element value_type;
  typedef Element* iterator;
  typedef const Element* const_iterator;

  // Constructs from a native array. References the source.
  NativeArray(const Element* array, size_t count, RelationToSourceReference) {
    InitRef(array, count);
  }

  // Constructs from a native array. Copies the source.
  NativeArray(const Element* array, size_t count, RelationToSourceCopy) {
    InitCopy(array, count);
  }

  // Copy constructor.
  NativeArray(const NativeArray& rhs) {
    (this->*rhs.clone_)(rhs.array_, rhs.size_);
  }

  ~NativeArray() {
    if (clone_ != &NativeArray::InitRef)
      delete[] array_;
  }

  // STL-style container methods.
  size_t size() const { return size_; }
  const_iterator begin() const { return array_; }
  const_iterator end() const { return array_ + size_; }
  bool operator==(const NativeArray& rhs) const {
    return size() == rhs.size() &&
        ArrayEq(begin(), size(), rhs.begin());
  }

 private:
  enum {
    kCheckTypeIsNotConstOrAReference = StaticAssertTypeEqHelper<
        Element, GTEST_REMOVE_REFERENCE_AND_CONST_(Element)>::value
  };

  // Initializes this object with a copy of the input.
  void InitCopy(const Element* array, size_t a_size) {
    Element* const copy = new Element[a_size];
    CopyArray(array, a_size, copy);
    array_ = copy;
    size_ = a_size;
    clone_ = &NativeArray::InitCopy;
  }

  // Initializes this object with a reference of the input.
  void InitRef(const Element* array, size_t a_size) {
    array_ = array;
    size_ = a_size;
    clone_ = &NativeArray::InitRef;
  }

  const Element* array_;
  size_t size_;
  void (NativeArray::*clone_)(const Element*, size_t);

  GTEST_DISALLOW_ASSIGN_(NativeArray);
};

}  // namespace internal
}  // namespace testing

#define GTEST_MESSAGE_AT_(file, line, message, result_type) \
  ::testing::internal::AssertHelper(result_type, file, line, message) \
    = ::testing::Message()

#define GTEST_MESSAGE_(message, result_type) \
  GTEST_MESSAGE_AT_(__FILE__, __LINE__, message, result_type)

#define GTEST_FATAL_FAILURE_(message) \
  return GTEST_MESSAGE_(message, ::testing::TestPartResult::kFatalFailure)

#define GTEST_NONFATAL_FAILURE_(message) \
  GTEST_MESSAGE_(message, ::testing::TestPartResult::kNonFatalFailure)

#define GTEST_SUCCESS_(message) \
  GTEST_MESSAGE_(message, ::testing::TestPartResult::kSuccess)

// Suppress MSVC warning 4702 (unreachable code) for the code following
// statement if it returns or throws (or doesn't return or throw in some
// situations).
#define GTEST_SUPPRESS_UNREACHABLE_CODE_WARNING_BELOW_(statement) \
  if (::testing::internal::AlwaysTrue()) { statement; }

#define GTEST_TEST_THROW_(statement, expected_exception, fail) \
  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)

#define GTEST_TEST_NO_THROW_(statement, fail) \
  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.")

#define GTEST_TEST_ANY_THROW_(statement, fail) \
  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.")


// Implements Boolean test assertions such as EXPECT_TRUE. expression can be
// either a boolean expression or an AssertionResult. text is a textual
// represenation of expression as it was passed into the EXPECT_TRUE.
#define GTEST_TEST_BOOLEAN_(expression, text, actual, expected, fail) \
  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())

#define GTEST_TEST_NO_FATAL_FAILURE_(statement, fail) \
  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.")

// Expands to the name of the class that implements the given test.
#define GTEST_TEST_CLASS_NAME_(test_case_name, test_name) \
  test_case_name##_##test_name##_Test

// Helper macro for defining tests.
#define GTEST_TEST_(test_case_name, test_name, parent_class, parent_id)\
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<\
            GTEST_TEST_CLASS_NAME_(test_case_name, test_name)>);\
void GTEST_TEST_CLASS_NAME_(test_case_name, test_name)::TestBody()

#endif  // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_INTERNAL_H_
```

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

```
#include <stdlib.h>
#include <assert.h>
#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.

```
// Copyright 2003 Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// A "smart" pointer type with reference tracking.  Every pointer to a
// particular object is kept on a circular linked list.  When the last pointer
// to an object is destroyed or reassigned, the object is deleted.
//
// Used properly, this deletes the object when the last reference goes away.
// There are several caveats:
// - Like all reference counting schemes, cycles lead to leaks.
// - Each smart pointer is actually two pointers (8 bytes instead of 4).
// - Every time a pointer is assigned, the entire list of pointers to that
//   object is traversed.  This class is therefore NOT SUITABLE when there
//   will often be more than two or three pointers to a particular object.
// - References are only tracked as long as linked_ptr<> objects are copied.
//   If a linked_ptr<> is converted to a raw pointer and back, BAD THINGS
//   will happen (double deletion).
//
// A good use of this class is storing object references in STL containers.
// You can safely put linked_ptr<> in a vector<>.
// Other uses may not be as good.
//
// Note: If you use an incomplete type with linked_ptr<>, the class
// *containing* linked_ptr<> must have a constructor and destructor (even
// if they do nothing!).
//
// Bill Gibbons suggested we use something like this.
//
// Thread Safety:
//   Unlike other linked_ptr implementations, in this implementation
//   a linked_ptr object is thread-safe in the sense that:
//     - it's safe to copy linked_ptr objects concurrently,
//     - it's safe to copy *from* a linked_ptr and read its underlying
//       raw pointer (e.g. via get()) concurrently, and
//     - it's safe to write to two linked_ptrs that point to the same
//       shared object concurrently.
// FIXME: rename this to safe_linked_ptr to avoid
// confusion with normal linked_ptr.

// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_LINKED_PTR_H_
#define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_LINKED_PTR_H_

#include <stdlib.h>
#include <assert.h>

#include "gtest/internal/gtest-port.h"

namespace testing {
namespace internal {

// Protects copying of all linked_ptr objects.
GTEST_API_ GTEST_DECLARE_STATIC_MUTEX_(g_linked_ptr_mutex);

// This is used internally by all instances of linked_ptr<>.  It needs to be
// a non-template class because different types of linked_ptr<> can refer to
// the same object (linked_ptr<Superclass>(obj) vs linked_ptr<Subclass>(obj)).
// So, it needs to be possible for different types of linked_ptr to participate
// in the same circular linked list, so we need a single class type here.
//
// DO NOT USE THIS CLASS DIRECTLY YOURSELF.  Use linked_ptr<T>.
class linked_ptr_internal {
 public:
  // Create a new circle that includes only this instance.
  void join_new() {
    next_ = this;
  }

  // Many linked_ptr operations may change p.link_ for some linked_ptr
  // variable p in the same circle as this object.  Therefore we need
  // to prevent two such operations from occurring concurrently.
  //
  // Note that different types of linked_ptr objects can coexist in a
  // circle (e.g. linked_ptr<Base>, linked_ptr<Derived1>, and
  // linked_ptr<Derived2>).  Therefore we must use a single mutex to
  // protect all linked_ptr objects.  This can create serious
  // contention in production code, but is acceptable in a testing
  // framework.

  // Join an existing circle.
  void join(linked_ptr_internal const* ptr)
      GTEST_LOCK_EXCLUDED_(g_linked_ptr_mutex) {
    MutexLock lock(&g_linked_ptr_mutex);

    linked_ptr_internal const* p = ptr;
    while (p->next_ != ptr) {
      assert(p->next_ != this &&
             "Trying to join() a linked ring we are already in. "
             "Is GMock thread safety enabled?");
      p = p->next_;
    }
    p->next_ = this;
    next_ = ptr;
  }

  // Leave whatever circle we're part of.  Returns true if we were the
  // last member of the circle.  Once this is done, you can join() another.
  bool depart()
      GTEST_LOCK_EXCLUDED_(g_linked_ptr_mutex) {
    MutexLock lock(&g_linked_ptr_mutex);

    if (next_ == this) return true;
    linked_ptr_internal const* p = next_;
    while (p->next_ != this) {
      assert(p->next_ != next_ &&
             "Trying to depart() a linked ring we are not in. "
             "Is GMock thread safety enabled?");
      p = p->next_;
    }
    p->next_ = next_;
    return false;
  }

 private:
  mutable linked_ptr_internal const* next_;
};

template <typename T>
class linked_ptr {
 public:
  typedef T element_type;

  // Take over ownership of a raw pointer.  This should happen as soon as
  // possible after the object is created.
  explicit linked_ptr(T* ptr = NULL) { capture(ptr); }
  ~linked_ptr() { depart(); }

  // Copy an existing linked_ptr<>, adding ourselves to the list of references.
  template <typename U> linked_ptr(linked_ptr<U> const& ptr) { copy(&ptr); }
  linked_ptr(linked_ptr const& ptr) {  // NOLINT
    assert(&ptr != this);
    copy(&ptr);
  }

  // Assignment releases the old value and acquires the new.
  template <typename U> linked_ptr& operator=(linked_ptr<U> const& ptr) {
    depart();
    copy(&ptr);
    return *this;
  }

  linked_ptr& operator=(linked_ptr const& ptr) {
    if (&ptr != this) {
      depart();
      copy(&ptr);
    }
    return *this;
  }

  // Smart pointer members.
  void reset(T* ptr = NULL) {
    depart();
    capture(ptr);
  }
  T* get() const { return value_; }
  T* operator->() const { return value_; }
  T& operator*() const { return *value_; }

  bool operator==(T* p) const { return value_ == p; }
  bool operator!=(T* p) const { return value_ != p; }
  template <typename U>
  bool operator==(linked_ptr<U> const& ptr) const {
    return value_ == ptr.get();
  }
  template <typename U>
  bool operator!=(linked_ptr<U> const& ptr) const {
    return value_ != ptr.get();
  }

 private:
  template <typename U>
  friend class linked_ptr;

  T* value_;
  linked_ptr_internal link_;

  void depart() {
    if (link_.depart()) delete value_;
  }

  void capture(T* ptr) {
    value_ = ptr;
    link_.join_new();
  }

  template <typename U> void copy(linked_ptr<U> const* ptr) {
    value_ = ptr->get();
    if (value_)
      link_.join(&ptr->link_);
    else
      link_.join_new();
  }
};

template<typename T> inline
bool operator==(T* ptr, const linked_ptr<T>& x) {
  return ptr == x.get();
}

template<typename T> inline
bool operator!=(T* ptr, const linked_ptr<T>& x) {
  return ptr != x.get();
}

// A function to convert T* into linked_ptr<T>
// Doing e.g. make_linked_ptr(new FooBarBaz<type>(arg)) is a shorter notation
// for linked_ptr<FooBarBaz<type> >(new FooBarBaz<type>(arg))
template <typename T>
linked_ptr<T> make_linked_ptr(T* ptr) {
  return linked_ptr<T>(ptr);
}

}  // namespace internal
}  // namespace testing

#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 >

- class testing::internal::ValueArray2< T1, T2 >
- class testing::internal::ValueArray3< T1, T2, T3 >
- class testing::internal::ValueArray4< T1, T2, T3, T4 >
- class testing::internal::ValueArray5< T1, T2, T3, T4, T5 >
- class testing::internal::ValueArray6< T1, T2, T3, T4, T5, T6 >
- class testing::internal::ValueArray7< T1, T2, T3, T4, T5, T6, T7 >
- class testing::internal::ValueArray8< T1, T2, T3, T4, T5, T6, T7, T8 >
- class testing::internal::ValueArray9< T1, T2, T3, T4, T5, T6, T7, T8, T9 >
- class testing::internal::ValueArray10< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 >
- class testing::internal::ValueArray11< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11 >
- class testing::internal::ValueArray12< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12 >
- class testing::internal::ValueArray13< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13 >
- class testing::internal::ValueArray14< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14 >
- class testing::internal::ValueArray15< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15 >
- class testing::internal::ValueArray16< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16 >
- class testing::internal::ValueArray17< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17 >
- class testing::internal::ValueArray18< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18 >
- class testing::internal::ValueArray19< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19 >
- class testing::internal::ValueArray20< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray21< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray22< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray23< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray24< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray25< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray26< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray27< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray28< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray29< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray30< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray31< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray32< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray33< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray34< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray35< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray36< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray37< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray38< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray39< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray40< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray41< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray42< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray43< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray44< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray45< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray46< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray47< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray48< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray49< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T
- class testing::internal::ValueArray50< T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T

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.

```
// This file was GENERATED by command:
//     pump.py gtest-param-util-generated.h.pump
// DO NOT EDIT BY HAND!!!

// Copyright 2008 Google Inc.
// All Rights Reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


// Type and function utilities for implementing parameterized tests.
// This file is generated by a SCRIPT.  DO NOT EDIT BY HAND!
//
// Currently Google Test supports at most 50 arguments in Values,
// and at most 10 arguments in Combine. Please contact
// googletestframework@googlegroups.com if you need more.
// Please note that the number of arguments to Combine is limited
// by the maximum arity of the implementation of tuple which is
// currently set at 10.

// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PARAM_UTIL_GENERATED_H_
#define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PARAM_UTIL_GENERATED_H_

#include "gtest/internal/gtest-param-util.h"
#include "gtest/internal/gtest-port.h"

namespace testing {

// Forward declarations of ValuesIn(), which is implemented in
// include/gtest/gtest-param-test.h.
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);

namespace internal {

// Used in the Values() function to provide polymorphic capabilities.
template <typename T1>
class ValueArray1 {
 public:
  explicit ValueArray1(T1 v1) : v1_(v1) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_)};
    return ValuesIn(array);
  }

  ValueArray1(const ValueArray1& other) : v1_(other.v1_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray1& other);

  const T1 v1_;
};

template <typename T1, typename T2>
class ValueArray2 {
 public:
  ValueArray2(T1 v1, T2 v2) : v1_(v1), v2_(v2) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_)};
    return ValuesIn(array);
  }

  ValueArray2(const ValueArray2& other) : v1_(other.v1_), v2_(other.v2_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray2& other);

  const T1 v1_;
  const T2 v2_;
};

template <typename T1, typename T2, typename T3>
class ValueArray3 {
 public:
  ValueArray3(T1 v1, T2 v2, T3 v3) : v1_(v1), v2_(v2), v3_(v3) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_)};
    return ValuesIn(array);
  }

  ValueArray3(const ValueArray3& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray3& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
};

template <typename T1, typename T2, typename T3, typename T4>
class ValueArray4 {
 public:
  ValueArray4(T1 v1, T2 v2, T3 v3, T4 v4) : v1_(v1), v2_(v2), v3_(v3),
      v4_(v4) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_)};
    return ValuesIn(array);
  }

  ValueArray4(const ValueArray4& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray4& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
};

template <typename T1, typename T2, typename T3, typename T4, typename T5>
class ValueArray5 {
 public:
  ValueArray5(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5) : v1_(v1), v2_(v2), v3_(v3),
      v4_(v4), v5_(v5) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_)};
    return ValuesIn(array);
  }

  ValueArray5(const ValueArray5& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray5& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
};

template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6>
class ValueArray6 {
 public:
  ValueArray6(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6) : v1_(v1), v2_(v2),
      v3_(v3), v4_(v4), v5_(v5), v6_(v6) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_)};
    return ValuesIn(array);
  }

  ValueArray6(const ValueArray6& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray6& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
};

template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7>
class ValueArray7 {
 public:
  ValueArray7(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7) : v1_(v1),
      v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_)};
    return ValuesIn(array);
  }

  ValueArray7(const ValueArray7& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray7& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
};

template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7, typename T8>
class ValueArray8 {
 public:
  ValueArray8(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7,
      T8 v8) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
      v8_(v8) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_)};
    return ValuesIn(array);
  }

  ValueArray8(const ValueArray8& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray8& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
};

template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7, typename T8, typename T9>
class ValueArray9 {
 public:
  ValueArray9(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8,
      T9 v9) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
      v8_(v8), v9_(v9) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_)};
    return ValuesIn(array);
  }

  ValueArray9(const ValueArray9& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray9& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
};

template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7, typename T8, typename T9, typename T10>
```

```
class ValueArray10 {
 public:
  ValueArray10(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
      T10 v10) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
      v8_(v8), v9_(v9), v10_(v10) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_)};
    return ValuesIn(array);
  }

  ValueArray10(const ValueArray10& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray10& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const 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>
class ValueArray11 {
 public:
  ValueArray11(T1 v1, T2 v2, T3 v3, T4 v4, T5 v5, T6 v6, T7 v7, T8 v8, T9 v9,
      T10 v10, T11 v11) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6),
      v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_)};
    return ValuesIn(array);
  }

  ValueArray11(const ValueArray11& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray11& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const 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>
class ValueArray12 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5),
      v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_)};
    return ValuesIn(array);
  }

  ValueArray12(const ValueArray12& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray12& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const 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>
class ValueArray13 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4),
      v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11),
      v12_(v12), v13_(v13) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_)};
    return ValuesIn(array);
  }

  ValueArray13(const ValueArray13& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray13& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const 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>
class ValueArray14 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3),
      v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
      v11_(v11), v12_(v12), v13_(v13), v14_(v14) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_)};
    return ValuesIn(array);
  }

  ValueArray14(const ValueArray14& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray14& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
};

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>
class ValueArray15 {
 public:
  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) : v1_(v1), v2_(v2),
      v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
      v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_)};
    return ValuesIn(array);
  }

  ValueArray15(const ValueArray15& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray15& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
};

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>
class ValueArray16 {
 public:
  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) : v1_(v1),
      v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9),
      v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15),
      v16_(v16) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_)};
    return ValuesIn(array);
  }

  ValueArray16(const ValueArray16& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray16& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
};

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>
class ValueArray17 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
      v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
      v15_(v15), v16_(v16), v17_(v17) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_)};
    return ValuesIn(array);
  }

  ValueArray17(const ValueArray17& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray17& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
};

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>
class ValueArray18 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
      v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
      v15_(v15), v16_(v16), v17_(v17), v18_(v18) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_)};
    return ValuesIn(array);
  }

  ValueArray18(const ValueArray18& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray18& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
};

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>
class ValueArray19 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6),
      v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13),
      v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_)};
    return ValuesIn(array);
  }

  ValueArray19(const ValueArray19& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray19& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
};

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>
class ValueArray20 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5),
      v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12),
      v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18),
      v19_(v19), v20_(v20) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_)};
    return ValuesIn(array);
  }

  ValueArray20(const ValueArray20& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray20& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
};

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>
class ValueArray21 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4),
      v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11),
      v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17),
      v18_(v18), v19_(v19), v20_(v20), v21_(v21) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_)};
    return ValuesIn(array);
  }

  ValueArray21(const ValueArray21& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray21& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
};

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>
class ValueArray22 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3),
      v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
      v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
      v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_)};
    return ValuesIn(array);
  }

  ValueArray22(const ValueArray22& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray22& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
};

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>
class ValueArray23 {
 public:
  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) : v1_(v1), v2_(v2),
      v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
      v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
      v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22),
      v23_(v23) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_)};
    return ValuesIn(array);
  }

  ValueArray23(const ValueArray23& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray23& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
};

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>
class ValueArray24 {
 public:
  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) : v1_(v1),
      v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9),
      v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15),
      v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21),
      v22_(v22), v23_(v23), v24_(v24) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_)};
    return ValuesIn(array);
  }

  ValueArray24(const ValueArray24& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray24& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
};

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>
class ValueArray25 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
      v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
      v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
      v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_)};
    return ValuesIn(array);
  }

  ValueArray25(const ValueArray25& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray25& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
};

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>
class ValueArray26 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
      v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
      v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
      v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_)};
    return ValuesIn(array);
  }

  ValueArray26(const ValueArray26& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray26& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
};

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>
class ValueArray27 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6),
      v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13),
      v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19),
      v20_(v20), v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25),
      v26_(v26), v27_(v27) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_)};
    return ValuesIn(array);
  }

  ValueArray27(const ValueArray27& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray27& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
};

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>
class ValueArray28 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5),
      v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12),
      v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18),
      v19_(v19), v20_(v20), v21_(v21), v22_(v22), v23_(v23), v24_(v24),
      v25_(v25), v26_(v26), v27_(v27), v28_(v28) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_)};
    return ValuesIn(array);
  }

  ValueArray28(const ValueArray28& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray28& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
};

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>
class ValueArray29 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4),
      v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11),
      v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17),
      v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22), v23_(v23),
      v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28), v29_(v29) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_)};
    return ValuesIn(array);
  }

  ValueArray29(const ValueArray29& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray29& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
};

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>
class ValueArray30 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3),
      v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
      v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
      v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22),
      v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28),
      v29_(v29), v30_(v30) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_)};
    return ValuesIn(array);
  }

  ValueArray30(const ValueArray30& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray30& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
};

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>
class ValueArray31 {
 public:
  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) : v1_(v1), v2_(v2),
      v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
      v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
      v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22),
      v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28),
      v29_(v29), v30_(v30), v31_(v31) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_)};
    return ValuesIn(array);
  }

  ValueArray31(const ValueArray31& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray31& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
};

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>
class ValueArray32 {
 public:
  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) : v1_(v1),
      v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9),
      v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15),
      v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21),
      v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27),
      v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_)};
    return ValuesIn(array);
  }

  ValueArray32(const ValueArray32& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray32& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
};

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>
class ValueArray33 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
      v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
      v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
      v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26),
      v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32),
      v33_(v33) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_)};
    return ValuesIn(array);
  }

  ValueArray33(const ValueArray33& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray33& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
};

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>
class ValueArray34 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
      v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
      v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
      v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26),
      v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32),
      v33_(v33), v34_(v34) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_)};
    return ValuesIn(array);
  }

  ValueArray34(const ValueArray34& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray34& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
};

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>
class ValueArray35 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6),
      v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13),
      v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19),
      v20_(v20), v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25),
      v26_(v26), v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31),
      v32_(v32), v33_(v33), v34_(v34), v35_(v35) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_)};
    return ValuesIn(array);
  }

  ValueArray35(const ValueArray35& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray35& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
};

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>
class ValueArray36 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5),
      v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12),
      v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18),
      v19_(v19), v20_(v20), v21_(v21), v22_(v22), v23_(v23), v24_(v24),
      v25_(v25), v26_(v26), v27_(v27), v28_(v28), v29_(v29), v30_(v30),
      v31_(v31), v32_(v32), v33_(v33), v34_(v34), v35_(v35), v36_(v36) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
        static_cast<T>(v36_)};
    return ValuesIn(array);
  }

  ValueArray36(const ValueArray36& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_), v36_(other.v36_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray36& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
  const T36 v36_;
};
```

```

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>
class ValueArray37 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4),
      v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11),
      v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17),
      v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22), v23_(v23),
      v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28), v29_(v29),
      v30_(v30), v31_(v31), v32_(v32), v33_(v33), v34_(v34), v35_(v35),
      v36_(v36), v37_(v37) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
        static_cast<T>(v36_), static_cast<T>(v37_)};
    return ValuesIn(array);
  }

  ValueArray37(const ValueArray37& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_), v36_(other.v36_), v37_(other.v37_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray37& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
  const T36 v36_;
  const T37 v37_;
};

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>
class ValueArray38 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3),
      v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
      v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
      v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22),
      v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28),
      v29_(v29), v30_(v30), v31_(v31), v32_(v32), v33_(v33), v34_(v34),
      v35_(v35), v36_(v36), v37_(v37), v38_(v38) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
        static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_)};
    return ValuesIn(array);
  }

  ValueArray38(const ValueArray38& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray38& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
  const T36 v36_;
  const T37 v37_;
  const T38 v38_;
};

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>
class ValueArray39 {
 public:
  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) : v1_(v1), v2_(v2),
      v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
      v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
      v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22),
      v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28),
      v29_(v29), v30_(v30), v31_(v31), v32_(v32), v33_(v33), v34_(v34),
      v35_(v35), v36_(v36), v37_(v37), v38_(v38), v39_(v39) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
        static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
        static_cast<T>(v39_)};
    return ValuesIn(array);
  }

  ValueArray39(const ValueArray39& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
      v39_(other.v39_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray39& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
  const T36 v36_;
  const T37 v37_;
  const T38 v38_;
  const T39 v39_;
};

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>
class ValueArray40 {
 public:
  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) : v1_(v1),
      v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9),
      v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15),
      v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21),
      v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27),
      v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32), v33_(v33),
      v34_(v34), v35_(v35), v36_(v36), v37_(v37), v38_(v38), v39_(v39),
      v40_(v40) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
        static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
        static_cast<T>(v39_), static_cast<T>(v40_)};
    return ValuesIn(array);
  }

  ValueArray40(const ValueArray40& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
      v39_(other.v39_), v40_(other.v40_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray40& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
  const T36 v36_;
  const T37 v37_;
  const T38 v38_;
  const T39 v39_;
  const T40 v40_;
};

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>
class ValueArray41 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
      v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
      v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
      v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26),
      v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32),
      v33_(v33), v34_(v34), v35_(v35), v36_(v36), v37_(v37), v38_(v38),
      v39_(v39), v40_(v40), v41_(v41) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
        static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
        static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_)};
    return ValuesIn(array);
  }

  ValueArray41(const ValueArray41& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
      v39_(other.v39_), v40_(other.v40_), v41_(other.v41_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray41& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
  const T36 v36_;
  const T37 v37_;
  const T38 v38_;
  const T39 v39_;
  const T40 v40_;
  const T41 v41_;
};

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>
class ValueArray42 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
      v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
      v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
      v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26),
      v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32),
      v33_(v33), v34_(v34), v35_(v35), v36_(v36), v37_(v37), v38_(v38),
      v39_(v39), v40_(v40), v41_(v41), v42_(v42) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
        static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
        static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
        static_cast<T>(v42_)};
    return ValuesIn(array);
  }

  ValueArray42(const ValueArray42& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
      v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray42& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
  const T36 v36_;
  const T37 v37_;
  const T38 v38_;
  const T39 v39_;
  const T40 v40_;
  const T41 v41_;
  const T42 v42_;
};

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>
class ValueArray43 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6),
      v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13),
      v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19),
      v20_(v20), v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25),
      v26_(v26), v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31),
      v32_(v32), v33_(v33), v34_(v34), v35_(v35), v36_(v36), v37_(v37),
      v38_(v38), v39_(v39), v40_(v40), v41_(v41), v42_(v42), v43_(v43) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
        static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
        static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
        static_cast<T>(v42_), static_cast<T>(v43_)};
    return ValuesIn(array);
  }

  ValueArray43(const ValueArray43& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
      v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
      v43_(other.v43_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray43& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
  const T36 v36_;
  const T37 v37_;
  const T38 v38_;
  const T39 v39_;
  const T40 v40_;
  const T41 v41_;
  const T42 v42_;
  const 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>
class ValueArray44 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5),
      v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12),
      v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17), v18_(v18),
      v19_(v19), v20_(v20), v21_(v21), v22_(v22), v23_(v23), v24_(v24),
      v25_(v25), v26_(v26), v27_(v27), v28_(v28), v29_(v29), v30_(v30),
      v31_(v31), v32_(v32), v33_(v33), v34_(v34), v35_(v35), v36_(v36),
      v37_(v37), v38_(v38), v39_(v39), v40_(v40), v41_(v41), v42_(v42),
      v43_(v43), v44_(v44) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
        static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
        static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
        static_cast<T>(v42_), static_cast<T>(v43_), static_cast<T>(v44_)};
    return ValuesIn(array);
  }

  ValueArray44(const ValueArray44& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
      v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
      v43_(other.v43_), v44_(other.v44_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray44& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
  const T36 v36_;
  const T37 v37_;
  const T38 v38_;
  const T39 v39_;
  const T40 v40_;
  const T41 v41_;
  const T42 v42_;
  const T43 v43_;
  const 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>
class ValueArray45 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4),
      v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10), v11_(v11),
      v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16), v17_(v17),
      v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22), v23_(v23),
      v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28), v29_(v29),
      v30_(v30), v31_(v31), v32_(v32), v33_(v33), v34_(v34), v35_(v35),
      v36_(v36), v37_(v37), v38_(v38), v39_(v39), v40_(v40), v41_(v41),
      v42_(v42), v43_(v43), v44_(v44), v45_(v45) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
        static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
        static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
        static_cast<T>(v42_), static_cast<T>(v43_), static_cast<T>(v44_),
        static_cast<T>(v45_)};
    return ValuesIn(array);
  }

  ValueArray45(const ValueArray45& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
      v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
      v43_(other.v43_), v44_(other.v44_), v45_(other.v45_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray45& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
  const T36 v36_;
  const T37 v37_;
  const T38 v38_;
  const T39 v39_;
  const T40 v40_;
  const T41 v41_;
  const T42 v42_;
  const T43 v43_;
  const T44 v44_;
  const 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>
class ValueArray46 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3),
      v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
      v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
      v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22),
      v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28),
      v29_(v29), v30_(v30), v31_(v31), v32_(v32), v33_(v33), v34_(v34),
      v35_(v35), v36_(v36), v37_(v37), v38_(v38), v39_(v39), v40_(v40),
      v41_(v41), v42_(v42), v43_(v43), v44_(v44), v45_(v45), v46_(v46) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
        static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
        static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
        static_cast<T>(v42_), static_cast<T>(v43_), static_cast<T>(v44_),
        static_cast<T>(v45_), static_cast<T>(v46_)};
    return ValuesIn(array);
  }

  ValueArray46(const ValueArray46& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
      v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
      v43_(other.v43_), v44_(other.v44_), v45_(other.v45_), v46_(other.v46_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray46& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
  const T36 v36_;
  const T37 v37_;
  const T38 v38_;
  const T39 v39_;
  const T40 v40_;
  const T41 v41_;
  const T42 v42_;
  const T43 v43_;
  const T44 v44_;
  const T45 v45_;
  const 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>
class ValueArray47 {
 public:
  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) : v1_(v1), v2_(v2),
      v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9), v10_(v10),
      v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15), v16_(v16),
      v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21), v22_(v22),
      v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27), v28_(v28),
      v29_(v29), v30_(v30), v31_(v31), v32_(v32), v33_(v33), v34_(v34),
      v35_(v35), v36_(v36), v37_(v37), v38_(v38), v39_(v39), v40_(v40),
      v41_(v41), v42_(v42), v43_(v43), v44_(v44), v45_(v45), v46_(v46),
      v47_(v47) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
        static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
        static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
        static_cast<T>(v42_), static_cast<T>(v43_), static_cast<T>(v44_),
        static_cast<T>(v45_), static_cast<T>(v46_), static_cast<T>(v47_)};
    return ValuesIn(array);
  }

  ValueArray47(const ValueArray47& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
      v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
      v43_(other.v43_), v44_(other.v44_), v45_(other.v45_), v46_(other.v46_),
      v47_(other.v47_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray47& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
  const T36 v36_;
  const T37 v37_;
  const T38 v38_;
  const T39 v39_;
  const T40 v40_;
  const T41 v41_;
  const T42 v42_;
  const T43 v43_;
  const T44 v44_;
  const T45 v45_;
  const T46 v46_;
  const 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>
class ValueArray48 {
 public:
  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) : v1_(v1),
      v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7), v8_(v8), v9_(v9),
      v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14), v15_(v15),
      v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20), v21_(v21),
      v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26), v27_(v27),
      v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32), v33_(v33),
      v34_(v34), v35_(v35), v36_(v36), v37_(v37), v38_(v38), v39_(v39),
      v40_(v40), v41_(v41), v42_(v42), v43_(v43), v44_(v44), v45_(v45),
      v46_(v46), v47_(v47), v48_(v48) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
        static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
        static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
        static_cast<T>(v42_), static_cast<T>(v43_), static_cast<T>(v44_),
        static_cast<T>(v45_), static_cast<T>(v46_), static_cast<T>(v47_),
        static_cast<T>(v48_)};
    return ValuesIn(array);
  }

  ValueArray48(const ValueArray48& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
      v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
      v43_(other.v43_), v44_(other.v44_), v45_(other.v45_), v46_(other.v46_),
      v47_(other.v47_), v48_(other.v48_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray48& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
  const T36 v36_;
  const T37 v37_;
  const T38 v38_;
  const T39 v39_;
  const T40 v40_;
  const T41 v41_;
  const T42 v42_;
  const T43 v43_;
  const T44 v44_;
  const T45 v45_;
  const T46 v46_;
  const T47 v47_;
  const 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>
class ValueArray49 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
      v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
      v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
      v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26),
      v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32),
      v33_(v33), v34_(v34), v35_(v35), v36_(v36), v37_(v37), v38_(v38),
      v39_(v39), v40_(v40), v41_(v41), v42_(v42), v43_(v43), v44_(v44),
      v45_(v45), v46_(v46), v47_(v47), v48_(v48), v49_(v49) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
        static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
        static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
        static_cast<T>(v42_), static_cast<T>(v43_), static_cast<T>(v44_),
        static_cast<T>(v45_), static_cast<T>(v46_), static_cast<T>(v47_),
        static_cast<T>(v48_), static_cast<T>(v49_)};
    return ValuesIn(array);
  }

  ValueArray49(const ValueArray49& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
      v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
      v43_(other.v43_), v44_(other.v44_), v45_(other.v45_), v46_(other.v46_),
      v47_(other.v47_), v48_(other.v48_), v49_(other.v49_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray49& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
  const T36 v36_;
  const T37 v37_;
  const T38 v38_;
  const T39 v39_;
  const T40 v40_;
  const T41 v41_;
  const T42 v42_;
  const T43 v43_;
  const T44 v44_;
  const T45 v45_;
  const T46 v46_;
  const T47 v47_;
  const T48 v48_;
  const 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>
class ValueArray50 {
 public:
  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) : v1_(v1), v2_(v2), v3_(v3), v4_(v4), v5_(v5), v6_(v6), v7_(v7),
      v8_(v8), v9_(v9), v10_(v10), v11_(v11), v12_(v12), v13_(v13), v14_(v14),
      v15_(v15), v16_(v16), v17_(v17), v18_(v18), v19_(v19), v20_(v20),
      v21_(v21), v22_(v22), v23_(v23), v24_(v24), v25_(v25), v26_(v26),
      v27_(v27), v28_(v28), v29_(v29), v30_(v30), v31_(v31), v32_(v32),
      v33_(v33), v34_(v34), v35_(v35), v36_(v36), v37_(v37), v38_(v38),
      v39_(v39), v40_(v40), v41_(v41), v42_(v42), v43_(v43), v44_(v44),
      v45_(v45), v46_(v46), v47_(v47), v48_(v48), v49_(v49), v50_(v50) {}

  template <typename T>
  operator ParamGenerator<T>() const {
    const T array[] = {static_cast<T>(v1_), static_cast<T>(v2_),
        static_cast<T>(v3_), static_cast<T>(v4_), static_cast<T>(v5_),
        static_cast<T>(v6_), static_cast<T>(v7_), static_cast<T>(v8_),
        static_cast<T>(v9_), static_cast<T>(v10_), static_cast<T>(v11_),
        static_cast<T>(v12_), static_cast<T>(v13_), static_cast<T>(v14_),
        static_cast<T>(v15_), static_cast<T>(v16_), static_cast<T>(v17_),
        static_cast<T>(v18_), static_cast<T>(v19_), static_cast<T>(v20_),
        static_cast<T>(v21_), static_cast<T>(v22_), static_cast<T>(v23_),
        static_cast<T>(v24_), static_cast<T>(v25_), static_cast<T>(v26_),
        static_cast<T>(v27_), static_cast<T>(v28_), static_cast<T>(v29_),
        static_cast<T>(v30_), static_cast<T>(v31_), static_cast<T>(v32_),
        static_cast<T>(v33_), static_cast<T>(v34_), static_cast<T>(v35_),
        static_cast<T>(v36_), static_cast<T>(v37_), static_cast<T>(v38_),
        static_cast<T>(v39_), static_cast<T>(v40_), static_cast<T>(v41_),
        static_cast<T>(v42_), static_cast<T>(v43_), static_cast<T>(v44_),
        static_cast<T>(v45_), static_cast<T>(v46_), static_cast<T>(v47_),
        static_cast<T>(v48_), static_cast<T>(v49_), static_cast<T>(v50_)};
    return ValuesIn(array);
  }

  ValueArray50(const ValueArray50& other) : v1_(other.v1_), v2_(other.v2_),
      v3_(other.v3_), v4_(other.v4_), v5_(other.v5_), v6_(other.v6_),
      v7_(other.v7_), v8_(other.v8_), v9_(other.v9_), v10_(other.v10_),
      v11_(other.v11_), v12_(other.v12_), v13_(other.v13_), v14_(other.v14_),
      v15_(other.v15_), v16_(other.v16_), v17_(other.v17_), v18_(other.v18_),
      v19_(other.v19_), v20_(other.v20_), v21_(other.v21_), v22_(other.v22_),
      v23_(other.v23_), v24_(other.v24_), v25_(other.v25_), v26_(other.v26_),
      v27_(other.v27_), v28_(other.v28_), v29_(other.v29_), v30_(other.v30_),
      v31_(other.v31_), v32_(other.v32_), v33_(other.v33_), v34_(other.v34_),
      v35_(other.v35_), v36_(other.v36_), v37_(other.v37_), v38_(other.v38_),
      v39_(other.v39_), v40_(other.v40_), v41_(other.v41_), v42_(other.v42_),
      v43_(other.v43_), v44_(other.v44_), v45_(other.v45_), v46_(other.v46_),
      v47_(other.v47_), v48_(other.v48_), v49_(other.v49_), v50_(other.v50_) {}

 private:
  // No implementation - assignment is unsupported.
  void operator=(const ValueArray50& other);

  const T1 v1_;
  const T2 v2_;
  const T3 v3_;
  const T4 v4_;
  const T5 v5_;
  const T6 v6_;
  const T7 v7_;
  const T8 v8_;
  const T9 v9_;
  const T10 v10_;
  const T11 v11_;
  const T12 v12_;
  const T13 v13_;
  const T14 v14_;
  const T15 v15_;
  const T16 v16_;
  const T17 v17_;
  const T18 v18_;
  const T19 v19_;
  const T20 v20_;
  const T21 v21_;
  const T22 v22_;
  const T23 v23_;
  const T24 v24_;
  const T25 v25_;
  const T26 v26_;
  const T27 v27_;
  const T28 v28_;
  const T29 v29_;
  const T30 v30_;
  const T31 v31_;
  const T32 v32_;
  const T33 v33_;
  const T34 v34_;
  const T35 v35_;
  const T36 v36_;
  const T37 v37_;
  const T38 v38_;
  const T39 v39_;
  const T40 v40_;
  const T41 v41_;
  const T42 v42_;
  const T43 v43_;
  const T44 v44_;
  const T45 v45_;
  const T46 v46_;
  const T47 v47_;
  const T48 v48_;
  const T49 v49_;
  const T50 v50_;
};

# if GTEST_HAS_COMBINE
// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
//
// Generates values from the Cartesian product of values produced
// by the argument generators.
//
template <typename T1, typename T2>
class CartesianProductGenerator2
    : public ParamGeneratorInterface< ::testing::tuple<T1, T2> > {
 public:
  typedef ::testing::tuple<T1, T2> ParamType;

  CartesianProductGenerator2(const ParamGenerator<T1>& g1,
      const ParamGenerator<T2>& g2)
      : g1_(g1), g2_(g2) {}
  virtual ~CartesianProductGenerator2() {}

  virtual ParamIteratorInterface<ParamType>* Begin() const {
    return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin());
  }
  virtual ParamIteratorInterface<ParamType>* End() const {
    return new Iterator(this, g1_, g1_.end(), g2_, g2_.end());
  }

 private:
  class Iterator : public ParamIteratorInterface<ParamType> {
   public:
    Iterator(const ParamGeneratorInterface<ParamType>* base,
      const ParamGenerator<T1>& g1,
      const typename ParamGenerator<T1>::iterator& current1,
      const ParamGenerator<T2>& g2,
      const typename ParamGenerator<T2>::iterator& current2)
        : base_(base),
          begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
          begin2_(g2.begin()), end2_(g2.end()), current2_(current2)    {
      ComputeCurrentValue();
    }
    virtual ~Iterator() {}

    virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
      return base_;
    }
    // Advance should not be called on beyond-of-range iterators
    // so no component iterators must be beyond end of range, either.
    virtual void Advance() {
      assert(!AtEnd());
      ++current2_;
      if (current2_ == end2_) {
        current2_ = begin2_;
        ++current1_;
      }
      ComputeCurrentValue();
    }
    virtual ParamIteratorInterface<ParamType>* Clone() const {
      return new Iterator(*this);
    }
    virtual const ParamType* Current() const { return current_value_.get(); }
    virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
      // Having the same base generator guarantees that the other
      // iterator is of the same type and we can downcast.
      GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
          « "The program attempted to compare iterators "
          « "from different generators." « std::endl;
      const Iterator* typed_other =
          CheckedDowncastToActualType<const Iterator>(&other);
      // We must report iterators equal if they both point beyond their
      // respective ranges. That can happen in a variety of fashions,
      // so we have to consult AtEnd().
      return (AtEnd() && typed_other->AtEnd()) ||
         (
          current1_ == typed_other->current1_ &&
          current2_ == typed_other->current2_);
    }

   private:
    Iterator(const Iterator& other)
        : base_(other.base_),
        begin1_(other.begin1_),
        end1_(other.end1_),
        current1_(other.current1_),
        begin2_(other.begin2_),
        end2_(other.end2_),
        current2_(other.current2_) {
      ComputeCurrentValue();
    }

    void ComputeCurrentValue() {
      if (!AtEnd())
        current_value_.reset(new ParamType(*current1_, *current2_));
    }
    bool AtEnd() const {
      // We must report iterator past the end of the range when either of the
      // component iterators has reached the end of its range.
      return
          current1_ == end1_ ||
          current2_ == end2_;
    }

    // No implementation - assignment is unsupported.
    void operator=(const Iterator& other);

    const ParamGeneratorInterface<ParamType>* const base_;
    // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
    // current[i]_ is the actual traversing iterator.
    const typename ParamGenerator<T1>::iterator begin1_;
    const typename ParamGenerator<T1>::iterator end1_;
    typename ParamGenerator<T1>::iterator current1_;
    const typename ParamGenerator<T2>::iterator begin2_;
    const typename ParamGenerator<T2>::iterator end2_;
    typename ParamGenerator<T2>::iterator current2_;
    linked_ptr<ParamType> current_value_;
  };  // class CartesianProductGenerator2::Iterator

  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductGenerator2& other);

  const ParamGenerator<T1> g1_;
  const ParamGenerator<T2> g2_;
};  // class CartesianProductGenerator2


template <typename T1, typename T2, typename T3>
class CartesianProductGenerator3
    : public ParamGeneratorInterface< ::testing::tuple<T1, T2, T3> > {
 public:
  typedef ::testing::tuple<T1, T2, T3> ParamType;

  CartesianProductGenerator3(const ParamGenerator<T1>& g1,
      const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3)
      : g1_(g1), g2_(g2), g3_(g3) {}
  virtual ~CartesianProductGenerator3() {}

  virtual ParamIteratorInterface<ParamType>* Begin() const {
    return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin(), g3_,
        g3_.begin());
  }
  virtual ParamIteratorInterface<ParamType>* End() const {
    return new Iterator(this, g1_, g1_.end(), g2_, g2_.end(), g3_, g3_.end());
  }

 private:
  class Iterator : public ParamIteratorInterface<ParamType> {
   public:
    Iterator(const ParamGeneratorInterface<ParamType>* base,
      const ParamGenerator<T1>& g1,
      const typename ParamGenerator<T1>::iterator& current1,
      const ParamGenerator<T2>& g2,
      const typename ParamGenerator<T2>::iterator& current2,
      const ParamGenerator<T3>& g3,
      const typename ParamGenerator<T3>::iterator& current3)
        : base_(base),
          begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
          begin2_(g2.begin()), end2_(g2.end()), current2_(current2),
          begin3_(g3.begin()), end3_(g3.end()), current3_(current3)    {
```

```
      ComputeCurrentValue();
    }
    virtual ~Iterator() {}

    virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
      return base_;
    }
    // Advance should not be called on beyond-of-range iterators
    // so no component iterators must be beyond end of range, either.
    virtual void Advance() {
      assert(!AtEnd());
      ++current3_;
      if (current3_ == end3_) {
        current3_ = begin3_;
        ++current2_;
      }
      if (current2_ == end2_) {
        current2_ = begin2_;
        ++current1_;
      }
      ComputeCurrentValue();
    }
    virtual ParamIteratorInterface<ParamType>* Clone() const {
      return new Iterator(*this);
    }
    virtual const ParamType* Current() const { return current_value_.get(); }
    virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
      // Having the same base generator guarantees that the other
      // iterator is of the same type and we can downcast.
      GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
          « "The program attempted to compare iterators "
          « "from different generators." « std::endl;
      const Iterator* typed_other =
          CheckedDowncastToActualType<const Iterator>(&other);
      // We must report iterators equal if they both point beyond their
      // respective ranges. That can happen in a variety of fashions,
      // so we have to consult AtEnd().
      return (AtEnd() && typed_other->AtEnd()) ||
         (
          current1_ == typed_other->current1_ &&
          current2_ == typed_other->current2_ &&
          current3_ == typed_other->current3_);
    }

   private:
    Iterator(const Iterator& other)
        : base_(other.base_),
        begin1_(other.begin1_),
        end1_(other.end1_),
        current1_(other.current1_),
        begin2_(other.begin2_),
        end2_(other.end2_),
        current2_(other.current2_),
        begin3_(other.begin3_),
        end3_(other.end3_),
        current3_(other.current3_) {
      ComputeCurrentValue();
    }

    void ComputeCurrentValue() {
      if (!AtEnd())
        current_value_.reset(new ParamType(*current1_, *current2_, *current3_));
    }
    bool AtEnd() const {
      // We must report iterator past the end of the range when either of the
      // component iterators has reached the end of its range.
      return
          current1_ == end1_ ||
          current2_ == end2_ ||
          current3_ == end3_;
    }

    // No implementation - assignment is unsupported.
    void operator=(const Iterator& other);

    const ParamGeneratorInterface<ParamType>* const base_;
    // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
    // current[i]_ is the actual traversing iterator.
    const typename ParamGenerator<T1>::iterator begin1_;
    const typename ParamGenerator<T1>::iterator end1_;
    typename ParamGenerator<T1>::iterator current1_;
    const typename ParamGenerator<T2>::iterator begin2_;
    const typename ParamGenerator<T2>::iterator end2_;
    typename ParamGenerator<T2>::iterator current2_;
    const typename ParamGenerator<T3>::iterator begin3_;
    const typename ParamGenerator<T3>::iterator end3_;
    typename ParamGenerator<T3>::iterator current3_;
    linked_ptr<ParamType> current_value_;
  };  // class CartesianProductGenerator3::Iterator

  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductGenerator3& other);

  const ParamGenerator<T1> g1_;
  const ParamGenerator<T2> g2_;
  const ParamGenerator<T3> g3_;
};  // class CartesianProductGenerator3


template <typename T1, typename T2, typename T3, typename T4>
class CartesianProductGenerator4
    : public ParamGeneratorInterface< ::testing::tuple<T1, T2, T3, T4> > {
 public:
  typedef ::testing::tuple<T1, T2, T3, T4> ParamType;

  CartesianProductGenerator4(const ParamGenerator<T1>& g1,
      const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3,
      const ParamGenerator<T4>& g4)
      : g1_(g1), g2_(g2), g3_(g3), g4_(g4) {}
  virtual ~CartesianProductGenerator4() {}

  virtual ParamIteratorInterface<ParamType>* Begin() const {
    return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin(), g3_,
        g3_.begin(), g4_, g4_.begin());
  }
  virtual ParamIteratorInterface<ParamType>* End() const {
    return new Iterator(this, g1_, g1_.end(), g2_, g2_.end(), g3_, g3_.end(),
        g4_, g4_.end());
  }

 private:
  class Iterator : public ParamIteratorInterface<ParamType> {
   public:
    Iterator(const ParamGeneratorInterface<ParamType>* base,
      const ParamGenerator<T1>& g1,
      const typename ParamGenerator<T1>::iterator& current1,
      const ParamGenerator<T2>& g2,
      const typename ParamGenerator<T2>::iterator& current2,
      const ParamGenerator<T3>& g3,
      const typename ParamGenerator<T3>::iterator& current3,
      const ParamGenerator<T4>& g4,
      const typename ParamGenerator<T4>::iterator& current4)
        : base_(base),
          begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
          begin2_(g2.begin()), end2_(g2.end()), current2_(current2),
          begin3_(g3.begin()), end3_(g3.end()), current3_(current3),
          begin4_(g4.begin()), end4_(g4.end()), current4_(current4)    {
      ComputeCurrentValue();
    }
    virtual ~Iterator() {}

    virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
      return base_;
    }
    // Advance should not be called on beyond-of-range iterators
    // so no component iterators must be beyond end of range, either.
    virtual void Advance() {
      assert(!AtEnd());
      ++current4_;
      if (current4_ == end4_) {
        current4_ = begin4_;
        ++current3_;
      }
      if (current3_ == end3_) {
        current3_ = begin3_;
        ++current2_;
      }
      if (current2_ == end2_) {
        current2_ = begin2_;
        ++current1_;
      }
      ComputeCurrentValue();
    }
    virtual ParamIteratorInterface<ParamType>* Clone() const {
      return new Iterator(*this);
    }
    virtual const ParamType* Current() const { return current_value_.get(); }
    virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
      // Having the same base generator guarantees that the other
      // iterator is of the same type and we can downcast.
      GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
          « "The program attempted to compare iterators "
          « "from different generators." « std::endl;
      const Iterator* typed_other =
          CheckedDowncastToActualType<const Iterator>(&other);
      // We must report iterators equal if they both point beyond their
      // respective ranges. That can happen in a variety of fashions,
      // so we have to consult AtEnd().
      return (AtEnd() && typed_other->AtEnd()) ||
         (
          current1_ == typed_other->current1_ &&
          current2_ == typed_other->current2_ &&
          current3_ == typed_other->current3_ &&
          current4_ == typed_other->current4_);
    }

   private:
    Iterator(const Iterator& other)
        : base_(other.base_),
        begin1_(other.begin1_),
        end1_(other.end1_),
        current1_(other.current1_),
        begin2_(other.begin2_),
        end2_(other.end2_),
        current2_(other.current2_),
        begin3_(other.begin3_),
        end3_(other.end3_),
        current3_(other.current3_),
        begin4_(other.begin4_),
        end4_(other.end4_),
        current4_(other.current4_) {
      ComputeCurrentValue();
    }

    void ComputeCurrentValue() {
      if (!AtEnd())
        current_value_.reset(new ParamType(*current1_, *current2_, *current3_,
            *current4_));
    }
    bool AtEnd() const {
      // We must report iterator past the end of the range when either of the
      // component iterators has reached the end of its range.
      return
          current1_ == end1_ ||
          current2_ == end2_ ||
          current3_ == end3_ ||
          current4_ == end4_;
    }

    // No implementation - assignment is unsupported.
    void operator=(const Iterator& other);

    const ParamGeneratorInterface<ParamType>* const base_;
    // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
    // current[i]_ is the actual traversing iterator.
    const typename ParamGenerator<T1>::iterator begin1_;
    const typename ParamGenerator<T1>::iterator end1_;
    typename ParamGenerator<T1>::iterator current1_;
    const typename ParamGenerator<T2>::iterator begin2_;
    const typename ParamGenerator<T2>::iterator end2_;
    typename ParamGenerator<T2>::iterator current2_;
    const typename ParamGenerator<T3>::iterator begin3_;
    const typename ParamGenerator<T3>::iterator end3_;
    typename ParamGenerator<T3>::iterator current3_;
    const typename ParamGenerator<T4>::iterator begin4_;
    const typename ParamGenerator<T4>::iterator end4_;
    typename ParamGenerator<T4>::iterator current4_;
    linked_ptr<ParamType> current_value_;
  };  // class CartesianProductGenerator4::Iterator

  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductGenerator4& other);

  const ParamGenerator<T1> g1_;
  const ParamGenerator<T2> g2_;
  const ParamGenerator<T3> g3_;
  const ParamGenerator<T4> g4_;
};  // class CartesianProductGenerator4


template <typename T1, typename T2, typename T3, typename T4, typename T5>
class CartesianProductGenerator5
    : public ParamGeneratorInterface< ::testing::tuple<T1, T2, T3, T4, T5> > {
 public:
  typedef ::testing::tuple<T1, T2, T3, T4, T5> ParamType;

  CartesianProductGenerator5(const ParamGenerator<T1>& g1,
      const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3,
      const ParamGenerator<T4>& g4, const ParamGenerator<T5>& g5)
      : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5) {}
  virtual ~CartesianProductGenerator5() {}

  virtual ParamIteratorInterface<ParamType>* Begin() const {
    return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin(), g3_,
        g3_.begin(), g4_, g4_.begin(), g5_, g5_.begin());
  }
  virtual ParamIteratorInterface<ParamType>* End() const {
    return new Iterator(this, g1_, g1_.end(), g2_, g2_.end(), g3_, g3_.end(),
        g4_, g4_.end(), g5_, g5_.end());
  }

 private:
  class Iterator : public ParamIteratorInterface<ParamType> {
   public:
    Iterator(const ParamGeneratorInterface<ParamType>* base,
      const ParamGenerator<T1>& g1,
      const typename ParamGenerator<T1>::iterator& current1,
      const ParamGenerator<T2>& g2,
      const typename ParamGenerator<T2>::iterator& current2,
      const ParamGenerator<T3>& g3,
      const typename ParamGenerator<T3>::iterator& current3,
      const ParamGenerator<T4>& g4,
      const typename ParamGenerator<T4>::iterator& current4,
      const ParamGenerator<T5>& g5,
      const typename ParamGenerator<T5>::iterator& current5)
        : base_(base),
          begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
          begin2_(g2.begin()), end2_(g2.end()), current2_(current2),
          begin3_(g3.begin()), end3_(g3.end()), current3_(current3),
          begin4_(g4.begin()), end4_(g4.end()), current4_(current4),
          begin5_(g5.begin()), end5_(g5.end()), current5_(current5)    {
      ComputeCurrentValue();
    }
    virtual ~Iterator() {}

    virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
      return base_;
    }
    // Advance should not be called on beyond-of-range iterators
    // so no component iterators must be beyond end of range, either.
    virtual void Advance() {
      assert(!AtEnd());
      ++current5_;
      if (current5_ == end5_) {
        current5_ = begin5_;
        ++current4_;
      }
      if (current4_ == end4_) {
        current4_ = begin4_;
        ++current3_;
      }
      if (current3_ == end3_) {
        current3_ = begin3_;
        ++current2_;
      }
      if (current2_ == end2_) {
        current2_ = begin2_;
        ++current1_;
      }
      ComputeCurrentValue();
    }
    virtual ParamIteratorInterface<ParamType>* Clone() const {
      return new Iterator(*this);
    }
    virtual const ParamType* Current() const { return current_value_.get(); }
    virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
      // Having the same base generator guarantees that the other
      // iterator is of the same type and we can downcast.
      GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
          « "The program attempted to compare iterators "
          « "from different generators." « std::endl;
      const Iterator* typed_other =
          CheckedDowncastToActualType<const Iterator>(&other);
      // We must report iterators equal if they both point beyond their
      // respective ranges. That can happen in a variety of fashions,
      // so we have to consult AtEnd().
      return (AtEnd() && typed_other->AtEnd()) ||
         (
          current1_ == typed_other->current1_ &&
          current2_ == typed_other->current2_ &&
          current3_ == typed_other->current3_ &&
          current4_ == typed_other->current4_ &&
          current5_ == typed_other->current5_);
    }

   private:
    Iterator(const Iterator& other)
        : base_(other.base_),
        begin1_(other.begin1_),
        end1_(other.end1_),
        current1_(other.current1_),
        begin2_(other.begin2_),
        end2_(other.end2_),
        current2_(other.current2_),
        begin3_(other.begin3_),
        end3_(other.end3_),
        current3_(other.current3_),
        begin4_(other.begin4_),
        end4_(other.end4_),
        current4_(other.current4_),
        begin5_(other.begin5_),
        end5_(other.end5_),
        current5_(other.current5_) {
      ComputeCurrentValue();
    }

    void ComputeCurrentValue() {
      if (!AtEnd())
        current_value_.reset(new ParamType(*current1_, *current2_, *current3_,
            *current4_, *current5_));
    }
    bool AtEnd() const {
      // We must report iterator past the end of the range when either of the
      // component iterators has reached the end of its range.
      return
          current1_ == end1_ ||
          current2_ == end2_ ||
          current3_ == end3_ ||
          current4_ == end4_ ||
          current5_ == end5_;
    }

    // No implementation - assignment is unsupported.
    void operator=(const Iterator& other);

    const ParamGeneratorInterface<ParamType>* const base_;
    // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
    // current[i]_ is the actual traversing iterator.
    const typename ParamGenerator<T1>::iterator begin1_;
    const typename ParamGenerator<T1>::iterator end1_;
    typename ParamGenerator<T1>::iterator current1_;
    const typename ParamGenerator<T2>::iterator begin2_;
    const typename ParamGenerator<T2>::iterator end2_;
    typename ParamGenerator<T2>::iterator current2_;
    const typename ParamGenerator<T3>::iterator begin3_;
    const typename ParamGenerator<T3>::iterator end3_;
    typename ParamGenerator<T3>::iterator current3_;
    const typename ParamGenerator<T4>::iterator begin4_;
    const typename ParamGenerator<T4>::iterator end4_;
    typename ParamGenerator<T4>::iterator current4_;
    const typename ParamGenerator<T5>::iterator begin5_;
    const typename ParamGenerator<T5>::iterator end5_;
    typename ParamGenerator<T5>::iterator current5_;
    linked_ptr<ParamType> current_value_;
  };  // class CartesianProductGenerator5::Iterator

  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductGenerator5& other);

  const ParamGenerator<T1> g1_;
  const ParamGenerator<T2> g2_;
  const ParamGenerator<T3> g3_;
  const ParamGenerator<T4> g4_;
  const ParamGenerator<T5> g5_;
};  // class CartesianProductGenerator5


template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6>
class CartesianProductGenerator6
    : public ParamGeneratorInterface< ::testing::tuple<T1, T2, T3, T4, T5,
        T6> > {
 public:
  typedef ::testing::tuple<T1, T2, T3, T4, T5, T6> ParamType;

  CartesianProductGenerator6(const ParamGenerator<T1>& g1,
      const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3,
      const ParamGenerator<T4>& g4, const ParamGenerator<T5>& g5,
      const ParamGenerator<T6>& g6)
      : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6) {}
  virtual ~CartesianProductGenerator6() {}

  virtual ParamIteratorInterface<ParamType>* Begin() const {
    return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin(), g3_,
        g3_.begin(), g4_, g4_.begin(), g5_, g5_.begin(), g6_, g6_.begin());
  }
  virtual ParamIteratorInterface<ParamType>* End() const {
    return new Iterator(this, g1_, g1_.end(), g2_, g2_.end(), g3_, g3_.end(),
        g4_, g4_.end(), g5_, g5_.end(), g6_, g6_.end());
  }

 private:
  class Iterator : public ParamIteratorInterface<ParamType> {
   public:
    Iterator(const ParamGeneratorInterface<ParamType>* base,
      const ParamGenerator<T1>& g1,
      const typename ParamGenerator<T1>::iterator& current1,
      const ParamGenerator<T2>& g2,
      const typename ParamGenerator<T2>::iterator& current2,
      const ParamGenerator<T3>& g3,
      const typename ParamGenerator<T3>::iterator& current3,
      const ParamGenerator<T4>& g4,
      const typename ParamGenerator<T4>::iterator& current4,
      const ParamGenerator<T5>& g5,
      const typename ParamGenerator<T5>::iterator& current5,
      const ParamGenerator<T6>& g6,
      const typename ParamGenerator<T6>::iterator& current6)
        : base_(base),
          begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
          begin2_(g2.begin()), end2_(g2.end()), current2_(current2),
          begin3_(g3.begin()), end3_(g3.end()), current3_(current3),
          begin4_(g4.begin()), end4_(g4.end()), current4_(current4),
          begin5_(g5.begin()), end5_(g5.end()), current5_(current5),
          begin6_(g6.begin()), end6_(g6.end()), current6_(current6)    {
      ComputeCurrentValue();
    }
    virtual ~Iterator() {}

    virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
      return base_;
    }
    // Advance should not be called on beyond-of-range iterators
    // so no component iterators must be beyond end of range, either.
    virtual void Advance() {
      assert(!AtEnd());
      ++current6_;
      if (current6_ == end6_) {
        current6_ = begin6_;
        ++current5_;
      }
      if (current5_ == end5_) {
        current5_ = begin5_;
        ++current4_;
      }
      if (current4_ == end4_) {
        current4_ = begin4_;
        ++current3_;
      }
      if (current3_ == end3_) {
        current3_ = begin3_;
        ++current2_;
      }
      if (current2_ == end2_) {
        current2_ = begin2_;
        ++current1_;
      }
      ComputeCurrentValue();
    }
    virtual ParamIteratorInterface<ParamType>* Clone() const {
      return new Iterator(*this);
    }
    virtual const ParamType* Current() const { return current_value_.get(); }
    virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
      // Having the same base generator guarantees that the other
      // iterator is of the same type and we can downcast.
      GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
          « "The program attempted to compare iterators "
          « "from different generators." « std::endl;
      const Iterator* typed_other =
          CheckedDowncastToActualType<const Iterator>(&other);
      // We must report iterators equal if they both point beyond their
      // respective ranges. That can happen in a variety of fashions,
      // so we have to consult AtEnd().
      return (AtEnd() && typed_other->AtEnd()) ||
         (
          current1_ == typed_other->current1_ &&
          current2_ == typed_other->current2_ &&
          current3_ == typed_other->current3_ &&
          current4_ == typed_other->current4_ &&
          current5_ == typed_other->current5_ &&
          current6_ == typed_other->current6_);
    }

   private:
    Iterator(const Iterator& other)
        : base_(other.base_),
        begin1_(other.begin1_),
        end1_(other.end1_),
        current1_(other.current1_),
        begin2_(other.begin2_),
        end2_(other.end2_),
        current2_(other.current2_),
        begin3_(other.begin3_),
        end3_(other.end3_),
        current3_(other.current3_),
        begin4_(other.begin4_),
        end4_(other.end4_),
        current4_(other.current4_),
        begin5_(other.begin5_),
        end5_(other.end5_),
        current5_(other.current5_),
        begin6_(other.begin6_),
        end6_(other.end6_),
        current6_(other.current6_) {
      ComputeCurrentValue();
    }

    void ComputeCurrentValue() {
      if (!AtEnd())
        current_value_.reset(new ParamType(*current1_, *current2_, *current3_,
            *current4_, *current5_, *current6_));
    }
    bool AtEnd() const {
      // We must report iterator past the end of the range when either of the
      // component iterators has reached the end of its range.
      return
          current1_ == end1_ ||
          current2_ == end2_ ||
          current3_ == end3_ ||
          current4_ == end4_ ||
          current5_ == end5_ ||
          current6_ == end6_;
    }

    // No implementation - assignment is unsupported.
    void operator=(const Iterator& other);

    const ParamGeneratorInterface<ParamType>* const base_;
    // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
    // current[i]_ is the actual traversing iterator.
    const typename ParamGenerator<T1>::iterator begin1_;
    const typename ParamGenerator<T1>::iterator end1_;
    typename ParamGenerator<T1>::iterator current1_;
    const typename ParamGenerator<T2>::iterator begin2_;
    const typename ParamGenerator<T2>::iterator end2_;
    typename ParamGenerator<T2>::iterator current2_;
    const typename ParamGenerator<T3>::iterator begin3_;
    const typename ParamGenerator<T3>::iterator end3_;
    typename ParamGenerator<T3>::iterator current3_;
    const typename ParamGenerator<T4>::iterator begin4_;
    const typename ParamGenerator<T4>::iterator end4_;
    typename ParamGenerator<T4>::iterator current4_;
    const typename ParamGenerator<T5>::iterator begin5_;
    const typename ParamGenerator<T5>::iterator end5_;
    typename ParamGenerator<T5>::iterator current5_;
    const typename ParamGenerator<T6>::iterator begin6_;
    const typename ParamGenerator<T6>::iterator end6_;
    typename ParamGenerator<T6>::iterator current6_;
    linked_ptr<ParamType> current_value_;
  };  // class CartesianProductGenerator6::Iterator

  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductGenerator6& other);

  const ParamGenerator<T1> g1_;
  const ParamGenerator<T2> g2_;
  const ParamGenerator<T3> g3_;
  const ParamGenerator<T4> g4_;
  const ParamGenerator<T5> g5_;
  const ParamGenerator<T6> g6_;
};  // class CartesianProductGenerator6


template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7>
class CartesianProductGenerator7
    : public ParamGeneratorInterface< ::testing::tuple<T1, T2, T3, T4, T5, T6,
        T7> > {
 public:
  typedef ::testing::tuple<T1, T2, T3, T4, T5, T6, T7> ParamType;

  CartesianProductGenerator7(const ParamGenerator<T1>& g1,
      const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3,
      const ParamGenerator<T4>& g4, const ParamGenerator<T5>& g5,
      const ParamGenerator<T6>& g6, const ParamGenerator<T7>& g7)
      : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7) {}
  virtual ~CartesianProductGenerator7() {}

  virtual ParamIteratorInterface<ParamType>* Begin() const {
    return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin(), g3_,
        g3_.begin(), g4_, g4_.begin(), g5_, g5_.begin(), g6_, g6_.begin(), g7_,
        g7_.begin());
  }
  virtual ParamIteratorInterface<ParamType>* End() const {
    return new Iterator(this, g1_, g1_.end(), g2_, g2_.end(), g3_, g3_.end(),
        g4_, g4_.end(), g5_, g5_.end(), g6_, g6_.end(), g7_, g7_.end());
  }

 private:
  class Iterator : public ParamIteratorInterface<ParamType> {
   public:
    Iterator(const ParamGeneratorInterface<ParamType>* base,
      const ParamGenerator<T1>& g1,
      const typename ParamGenerator<T1>::iterator& current1,
      const ParamGenerator<T2>& g2,
      const typename ParamGenerator<T2>::iterator& current2,
      const ParamGenerator<T3>& g3,
      const typename ParamGenerator<T3>::iterator& current3,
      const ParamGenerator<T4>& g4,
      const typename ParamGenerator<T4>::iterator& current4,
      const ParamGenerator<T5>& g5,
      const typename ParamGenerator<T5>::iterator& current5,
      const ParamGenerator<T6>& g6,
      const typename ParamGenerator<T6>::iterator& current6,
      const ParamGenerator<T7>& g7,
      const typename ParamGenerator<T7>::iterator& current7)
        : base_(base),
          begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
          begin2_(g2.begin()), end2_(g2.end()), current2_(current2),
          begin3_(g3.begin()), end3_(g3.end()), current3_(current3),
          begin4_(g4.begin()), end4_(g4.end()), current4_(current4),
          begin5_(g5.begin()), end5_(g5.end()), current5_(current5),
          begin6_(g6.begin()), end6_(g6.end()), current6_(current6),
          begin7_(g7.begin()), end7_(g7.end()), current7_(current7)    {
      ComputeCurrentValue();
    }
    virtual ~Iterator() {}

    virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
      return base_;
    }
    // Advance should not be called on beyond-of-range iterators
    // so no component iterators must be beyond end of range, either.
    virtual void Advance() {
      assert(!AtEnd());
      ++current7_;
      if (current7_ == end7_) {
        current7_ = begin7_;
        ++current6_;
      }
      if (current6_ == end6_) {
        current6_ = begin6_;
        ++current5_;
      }
      if (current5_ == end5_) {
        current5_ = begin5_;
        ++current4_;
      }
      if (current4_ == end4_) {
        current4_ = begin4_;
        ++current3_;
      }
      if (current3_ == end3_) {
        current3_ = begin3_;
        ++current2_;
      }
      if (current2_ == end2_) {
        current2_ = begin2_;
        ++current1_;
      }
      ComputeCurrentValue();
    }
    virtual ParamIteratorInterface<ParamType>* Clone() const {
      return new Iterator(*this);
    }
    virtual const ParamType* Current() const { return current_value_.get(); }
    virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
      // Having the same base generator guarantees that the other
      // iterator is of the same type and we can downcast.
      GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
          « "The program attempted to compare iterators "
          « "from different generators." « std::endl;
      const Iterator* typed_other =
          CheckedDowncastToActualType<const Iterator>(&other);
      // We must report iterators equal if they both point beyond their
      // respective ranges. That can happen in a variety of fashions,
      // so we have to consult AtEnd().
      return (AtEnd() && typed_other->AtEnd()) ||
         (
          current1_ == typed_other->current1_ &&
          current2_ == typed_other->current2_ &&
          current3_ == typed_other->current3_ &&
          current4_ == typed_other->current4_ &&
          current5_ == typed_other->current5_ &&
          current6_ == typed_other->current6_ &&
          current7_ == typed_other->current7_);
    }

   private:
    Iterator(const Iterator& other)
        : base_(other.base_),
        begin1_(other.begin1_),
        end1_(other.end1_),
        current1_(other.current1_),
        begin2_(other.begin2_),
        end2_(other.end2_),
        current2_(other.current2_),
        begin3_(other.begin3_),
        end3_(other.end3_),
        current3_(other.current3_),
        begin4_(other.begin4_),
        end4_(other.end4_),
        current4_(other.current4_),
        begin5_(other.begin5_),
        end5_(other.end5_),
        current5_(other.current5_),
        begin6_(other.begin6_),
        end6_(other.end6_),
        current6_(other.current6_),
        begin7_(other.begin7_),
        end7_(other.end7_),
        current7_(other.current7_) {
      ComputeCurrentValue();
    }

    void ComputeCurrentValue() {
      if (!AtEnd())
        current_value_.reset(new ParamType(*current1_, *current2_, *current3_,
            *current4_, *current5_, *current6_, *current7_));
    }
    bool AtEnd() const {
      // We must report iterator past the end of the range when either of the
      // component iterators has reached the end of its range.
      return
          current1_ == end1_ ||
          current2_ == end2_ ||
          current3_ == end3_ ||
          current4_ == end4_ ||
          current5_ == end5_ ||
          current6_ == end6_ ||
          current7_ == end7_;
    }

    // No implementation - assignment is unsupported.
    void operator=(const Iterator& other);

    const ParamGeneratorInterface<ParamType>* const base_;
    // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
    // current[i]_ is the actual traversing iterator.
    const typename ParamGenerator<T1>::iterator begin1_;
    const typename ParamGenerator<T1>::iterator end1_;
    typename ParamGenerator<T1>::iterator current1_;
    const typename ParamGenerator<T2>::iterator begin2_;
    const typename ParamGenerator<T2>::iterator end2_;
    typename ParamGenerator<T2>::iterator current2_;
    const typename ParamGenerator<T3>::iterator begin3_;
    const typename ParamGenerator<T3>::iterator end3_;
    typename ParamGenerator<T3>::iterator current3_;
    const typename ParamGenerator<T4>::iterator begin4_;
    const typename ParamGenerator<T4>::iterator end4_;
    typename ParamGenerator<T4>::iterator current4_;
    const typename ParamGenerator<T5>::iterator begin5_;
    const typename ParamGenerator<T5>::iterator end5_;
    typename ParamGenerator<T5>::iterator current5_;
    const typename ParamGenerator<T6>::iterator begin6_;
    const typename ParamGenerator<T6>::iterator end6_;
    typename ParamGenerator<T6>::iterator current6_;
    const typename ParamGenerator<T7>::iterator begin7_;
    const typename ParamGenerator<T7>::iterator end7_;
    typename ParamGenerator<T7>::iterator current7_;
    linked_ptr<ParamType> current_value_;
  };  // class CartesianProductGenerator7::Iterator

  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductGenerator7& other);

  const ParamGenerator<T1> g1_;
  const ParamGenerator<T2> g2_;
  const ParamGenerator<T3> g3_;
  const ParamGenerator<T4> g4_;
  const ParamGenerator<T5> g5_;
  const ParamGenerator<T6> g6_;
  const ParamGenerator<T7> g7_;
};  // class CartesianProductGenerator7


template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7, typename T8>
class CartesianProductGenerator8
    : public ParamGeneratorInterface< ::testing::tuple<T1, T2, T3, T4, T5, T6,
        T7, T8> > {
 public:
  typedef ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8> ParamType;

  CartesianProductGenerator8(const ParamGenerator<T1>& g1,
      const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3,
      const ParamGenerator<T4>& g4, const ParamGenerator<T5>& g5,
      const ParamGenerator<T6>& g6, const ParamGenerator<T7>& g7,
      const ParamGenerator<T8>& g8)
      : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7),
          g8_(g8) {}
  virtual ~CartesianProductGenerator8() {}

  virtual ParamIteratorInterface<ParamType>* Begin() const {
    return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin(), g3_,
        g3_.begin(), g4_, g4_.begin(), g5_, g5_.begin(), g6_, g6_.begin(), g7_,
        g7_.begin(), g8_, g8_.begin());
  }
  virtual ParamIteratorInterface<ParamType>* End() const {
    return new Iterator(this, g1_, g1_.end(), g2_, g2_.end(), g3_, g3_.end(),
        g4_, g4_.end(), g5_, g5_.end(), g6_, g6_.end(), g7_, g7_.end(), g8_,
        g8_.end());
  }

 private:
  class Iterator : public ParamIteratorInterface<ParamType> {
   public:
    Iterator(const ParamGeneratorInterface<ParamType>* base,
      const ParamGenerator<T1>& g1,
      const typename ParamGenerator<T1>::iterator& current1,
      const ParamGenerator<T2>& g2,
      const typename ParamGenerator<T2>::iterator& current2,
      const ParamGenerator<T3>& g3,
      const typename ParamGenerator<T3>::iterator& current3,
      const ParamGenerator<T4>& g4,
      const typename ParamGenerator<T4>::iterator& current4,
      const ParamGenerator<T5>& g5,
      const typename ParamGenerator<T5>::iterator& current5,
      const ParamGenerator<T6>& g6,
      const typename ParamGenerator<T6>::iterator& current6,
      const ParamGenerator<T7>& g7,
      const typename ParamGenerator<T7>::iterator& current7,
      const ParamGenerator<T8>& g8,
      const typename ParamGenerator<T8>::iterator& current8)
        : base_(base),
          begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
          begin2_(g2.begin()), end2_(g2.end()), current2_(current2),
          begin3_(g3.begin()), end3_(g3.end()), current3_(current3),
          begin4_(g4.begin()), end4_(g4.end()), current4_(current4),
          begin5_(g5.begin()), end5_(g5.end()), current5_(current5),
          begin6_(g6.begin()), end6_(g6.end()), current6_(current6),
          begin7_(g7.begin()), end7_(g7.end()), current7_(current7),
          begin8_(g8.begin()), end8_(g8.end()), current8_(current8)    {
      ComputeCurrentValue();
    }
    virtual ~Iterator() {}

    virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
      return base_;
    }
    // Advance should not be called on beyond-of-range iterators
    // so no component iterators must be beyond end of range, either.
    virtual void Advance() {
      assert(!AtEnd());
      ++current8_;
      if (current8_ == end8_) {
        current8_ = begin8_;
        ++current7_;
      }
      if (current7_ == end7_) {
        current7_ = begin7_;
        ++current6_;
      }
      if (current6_ == end6_) {
        current6_ = begin6_;
        ++current5_;
      }
      if (current5_ == end5_) {
        current5_ = begin5_;
        ++current4_;
      }
      if (current4_ == end4_) {
        current4_ = begin4_;
        ++current3_;
      }
      if (current3_ == end3_) {
        current3_ = begin3_;
        ++current2_;
      }
      if (current2_ == end2_) {
        current2_ = begin2_;
        ++current1_;
      }
      ComputeCurrentValue();
    }
    virtual ParamIteratorInterface<ParamType>* Clone() const {
      return new Iterator(*this);
    }
    virtual const ParamType* Current() const { return current_value_.get(); }
    virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
      // Having the same base generator guarantees that the other
      // iterator is of the same type and we can downcast.
      GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
          « "The program attempted to compare iterators "
          « "from different generators." « std::endl;
      const Iterator* typed_other =
          CheckedDowncastToActualType<const Iterator>(&other);
      // We must report iterators equal if they both point beyond their
      // respective ranges. That can happen in a variety of fashions,
      // so we have to consult AtEnd().
      return (AtEnd() && typed_other->AtEnd()) ||
         (
          current1_ == typed_other->current1_ &&
          current2_ == typed_other->current2_ &&
          current3_ == typed_other->current3_ &&
          current4_ == typed_other->current4_ &&
          current5_ == typed_other->current5_ &&
          current6_ == typed_other->current6_ &&
          current7_ == typed_other->current7_ &&
          current8_ == typed_other->current8_);
    }

   private:
    Iterator(const Iterator& other)
        : base_(other.base_),
        begin1_(other.begin1_),
        end1_(other.end1_),
        current1_(other.current1_),
        begin2_(other.begin2_),
        end2_(other.end2_),
        current2_(other.current2_),
        begin3_(other.begin3_),
        end3_(other.end3_),
        current3_(other.current3_),
        begin4_(other.begin4_),
        end4_(other.end4_),
        current4_(other.current4_),
        begin5_(other.begin5_),
        end5_(other.end5_),
        current5_(other.current5_),
        begin6_(other.begin6_),
        end6_(other.end6_),
        current6_(other.current6_),
        begin7_(other.begin7_),
        end7_(other.end7_),
        current7_(other.current7_),
        begin8_(other.begin8_),
        end8_(other.end8_),
        current8_(other.current8_) {
      ComputeCurrentValue();
    }

    void ComputeCurrentValue() {
      if (!AtEnd())
        current_value_.reset(new ParamType(*current1_, *current2_, *current3_,
            *current4_, *current5_, *current6_, *current7_, *current8_));
    }
    bool AtEnd() const {
      // We must report iterator past the end of the range when either of the
      // component iterators has reached the end of its range.
      return
          current1_ == end1_ ||
          current2_ == end2_ ||
          current3_ == end3_ ||
          current4_ == end4_ ||
          current5_ == end5_ ||
          current6_ == end6_ ||
          current7_ == end7_ ||
          current8_ == end8_;
    }

    // No implementation - assignment is unsupported.
    void operator=(const Iterator& other);

    const ParamGeneratorInterface<ParamType>* const base_;
    // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
    // current[i]_ is the actual traversing iterator.
    const typename ParamGenerator<T1>::iterator begin1_;
    const typename ParamGenerator<T1>::iterator end1_;
    typename ParamGenerator<T1>::iterator current1_;
    const typename ParamGenerator<T2>::iterator begin2_;
    const typename ParamGenerator<T2>::iterator end2_;
    typename ParamGenerator<T2>::iterator current2_;
    const typename ParamGenerator<T3>::iterator begin3_;
    const typename ParamGenerator<T3>::iterator end3_;
    typename ParamGenerator<T3>::iterator current3_;
    const typename ParamGenerator<T4>::iterator begin4_;
    const typename ParamGenerator<T4>::iterator end4_;
    typename ParamGenerator<T4>::iterator current4_;
    const typename ParamGenerator<T5>::iterator begin5_;
    const typename ParamGenerator<T5>::iterator end5_;
    typename ParamGenerator<T5>::iterator current5_;
    const typename ParamGenerator<T6>::iterator begin6_;
    const typename ParamGenerator<T6>::iterator end6_;
    typename ParamGenerator<T6>::iterator current6_;
    const typename ParamGenerator<T7>::iterator begin7_;
    const typename ParamGenerator<T7>::iterator end7_;
    typename ParamGenerator<T7>::iterator current7_;
    const typename ParamGenerator<T8>::iterator begin8_;
    const typename ParamGenerator<T8>::iterator end8_;
    typename ParamGenerator<T8>::iterator current8_;
    linked_ptr<ParamType> current_value_;
  };  // class CartesianProductGenerator8::Iterator

  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductGenerator8& other);

  const ParamGenerator<T1> g1_;
  const ParamGenerator<T2> g2_;
  const ParamGenerator<T3> g3_;
  const ParamGenerator<T4> g4_;
  const ParamGenerator<T5> g5_;
  const ParamGenerator<T6> g6_;
  const ParamGenerator<T7> g7_;
  const ParamGenerator<T8> g8_;
};  // class CartesianProductGenerator8


template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7, typename T8, typename T9>
class CartesianProductGenerator9
    : public ParamGeneratorInterface< ::testing::tuple<T1, T2, T3, T4, T5, T6,
        T7, T8, T9> > {
 public:
  typedef ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8, T9> ParamType;

  CartesianProductGenerator9(const ParamGenerator<T1>& g1,
      const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3,
      const ParamGenerator<T4>& g4, const ParamGenerator<T5>& g5,
      const ParamGenerator<T6>& g6, const ParamGenerator<T7>& g7,
      const ParamGenerator<T8>& g8, const ParamGenerator<T9>& g9)
      : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7), g8_(g8),
          g9_(g9) {}
  virtual ~CartesianProductGenerator9() {}

  virtual ParamIteratorInterface<ParamType>* Begin() const {
    return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin(), g3_,
        g3_.begin(), g4_, g4_.begin(), g5_, g5_.begin(), g6_, g6_.begin(), g7_,
        g7_.begin(), g8_, g8_.begin(), g9_, g9_.begin());
  }
  virtual ParamIteratorInterface<ParamType>* End() const {
    return new Iterator(this, g1_, g1_.end(), g2_, g2_.end(), g3_, g3_.end(),
        g4_, g4_.end(), g5_, g5_.end(), g6_, g6_.end(), g7_, g7_.end(), g8_,
        g8_.end(), g9_, g9_.end());
  }

 private:
  class Iterator : public ParamIteratorInterface<ParamType> {
   public:
    Iterator(const ParamGeneratorInterface<ParamType>* base,
      const ParamGenerator<T1>& g1,
      const typename ParamGenerator<T1>::iterator& current1,
      const ParamGenerator<T2>& g2,
      const typename ParamGenerator<T2>::iterator& current2,
      const ParamGenerator<T3>& g3,
      const typename ParamGenerator<T3>::iterator& current3,
      const ParamGenerator<T4>& g4,
      const typename ParamGenerator<T4>::iterator& current4,
      const ParamGenerator<T5>& g5,
      const typename ParamGenerator<T5>::iterator& current5,
      const ParamGenerator<T6>& g6,
      const typename ParamGenerator<T6>::iterator& current6,
      const ParamGenerator<T7>& g7,
      const typename ParamGenerator<T7>::iterator& current7,
      const ParamGenerator<T8>& g8,
      const typename ParamGenerator<T8>::iterator& current8,
      const ParamGenerator<T9>& g9,
      const typename ParamGenerator<T9>::iterator& current9)
        : base_(base),
          begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
          begin2_(g2.begin()), end2_(g2.end()), current2_(current2),
          begin3_(g3.begin()), end3_(g3.end()), current3_(current3),
          begin4_(g4.begin()), end4_(g4.end()), current4_(current4),
          begin5_(g5.begin()), end5_(g5.end()), current5_(current5),
          begin6_(g6.begin()), end6_(g6.end()), current6_(current6),
          begin7_(g7.begin()), end7_(g7.end()), current7_(current7),
          begin8_(g8.begin()), end8_(g8.end()), current8_(current8),
          begin9_(g9.begin()), end9_(g9.end()), current9_(current9)    {
      ComputeCurrentValue();
    }
    virtual ~Iterator() {}

    virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
      return base_;
    }
    // Advance should not be called on beyond-of-range iterators
    // so no component iterators must be beyond end of range, either.
    virtual void Advance() {
      assert(!AtEnd());
      ++current9_;
      if (current9_ == end9_) {
        current9_ = begin9_;
        ++current8_;
      }
      if (current8_ == end8_) {
        current8_ = begin8_;
        ++current7_;
      }
      if (current7_ == end7_) {
        current7_ = begin7_;
        ++current6_;
      }
      if (current6_ == end6_) {
        current6_ = begin6_;
        ++current5_;
      }
      if (current5_ == end5_) {
        current5_ = begin5_;
        ++current4_;
      }
      if (current4_ == end4_) {
        current4_ = begin4_;
        ++current3_;
      }
      if (current3_ == end3_) {
        current3_ = begin3_;
        ++current2_;
      }
      if (current2_ == end2_) {
        current2_ = begin2_;
        ++current1_;
      }
      ComputeCurrentValue();
    }
    virtual ParamIteratorInterface<ParamType>* Clone() const {
      return new Iterator(*this);
    }
    virtual const ParamType* Current() const { return current_value_.get(); }
    virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
      // Having the same base generator guarantees that the other
      // iterator is of the same type and we can downcast.
      GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
          « "The program attempted to compare iterators "
          « "from different generators." « std::endl;
      const Iterator* typed_other =
          CheckedDowncastToActualType<const Iterator>(&other);
      // We must report iterators equal if they both point beyond their
      // respective ranges. That can happen in a variety of fashions,
      // so we have to consult AtEnd().
      return (AtEnd() && typed_other->AtEnd()) ||
         (
          current1_ == typed_other->current1_ &&
          current2_ == typed_other->current2_ &&
          current3_ == typed_other->current3_ &&
          current4_ == typed_other->current4_ &&
          current5_ == typed_other->current5_ &&
          current6_ == typed_other->current6_ &&
          current7_ == typed_other->current7_ &&
          current8_ == typed_other->current8_ &&
          current9_ == typed_other->current9_);
    }

   private:
    Iterator(const Iterator& other)
        : base_(other.base_),
        begin1_(other.begin1_),
        end1_(other.end1_),
        current1_(other.current1_),
        begin2_(other.begin2_),
        end2_(other.end2_),
        current2_(other.current2_),
        begin3_(other.begin3_),
        end3_(other.end3_),
        current3_(other.current3_),
        begin4_(other.begin4_),
        end4_(other.end4_),
        current4_(other.current4_),
        begin5_(other.begin5_),
        end5_(other.end5_),
        current5_(other.current5_),
        begin6_(other.begin6_),
        end6_(other.end6_),
        current6_(other.current6_),
        begin7_(other.begin7_),
        end7_(other.end7_),
        current7_(other.current7_),
        begin8_(other.begin8_),
        end8_(other.end8_),
        current8_(other.current8_),
        begin9_(other.begin9_),
        end9_(other.end9_),
        current9_(other.current9_) {
      ComputeCurrentValue();
    }

    void ComputeCurrentValue() {
      if (!AtEnd())
        current_value_.reset(new ParamType(*current1_, *current2_, *current3_,
            *current4_, *current5_, *current6_, *current7_, *current8_,
            *current9_));
    }
    bool AtEnd() const {
      // We must report iterator past the end of the range when either of the
      // component iterators has reached the end of its range.
      return
          current1_ == end1_ ||
          current2_ == end2_ ||
          current3_ == end3_ ||
          current4_ == end4_ ||
          current5_ == end5_ ||
          current6_ == end6_ ||
          current7_ == end7_ ||
          current8_ == end8_ ||
          current9_ == end9_;
    }

    // No implementation - assignment is unsupported.
    void operator=(const Iterator& other);

    const ParamGeneratorInterface<ParamType>* const base_;
    // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
    // current[i]_ is the actual traversing iterator.
    const typename ParamGenerator<T1>::iterator begin1_;
    const typename ParamGenerator<T1>::iterator end1_;
    typename ParamGenerator<T1>::iterator current1_;
    const typename ParamGenerator<T2>::iterator begin2_;
    const typename ParamGenerator<T2>::iterator end2_;
    typename ParamGenerator<T2>::iterator current2_;
    const typename ParamGenerator<T3>::iterator begin3_;
    const typename ParamGenerator<T3>::iterator end3_;
    typename ParamGenerator<T3>::iterator current3_;
    const typename ParamGenerator<T4>::iterator begin4_;
    const typename ParamGenerator<T4>::iterator end4_;
    typename ParamGenerator<T4>::iterator current4_;
    const typename ParamGenerator<T5>::iterator begin5_;
    const typename ParamGenerator<T5>::iterator end5_;
    typename ParamGenerator<T5>::iterator current5_;
    const typename ParamGenerator<T6>::iterator begin6_;
    const typename ParamGenerator<T6>::iterator end6_;
    typename ParamGenerator<T6>::iterator current6_;
    const typename ParamGenerator<T7>::iterator begin7_;
    const typename ParamGenerator<T7>::iterator end7_;
    typename ParamGenerator<T7>::iterator current7_;
    const typename ParamGenerator<T8>::iterator begin8_;
    const typename ParamGenerator<T8>::iterator end8_;
    typename ParamGenerator<T8>::iterator current8_;
    const typename ParamGenerator<T9>::iterator begin9_;
    const typename ParamGenerator<T9>::iterator end9_;
    typename ParamGenerator<T9>::iterator current9_;
    linked_ptr<ParamType> current_value_;
  };  // class CartesianProductGenerator9::Iterator

  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductGenerator9& other);

  const ParamGenerator<T1> g1_;
  const ParamGenerator<T2> g2_;
  const ParamGenerator<T3> g3_;
  const ParamGenerator<T4> g4_;
  const ParamGenerator<T5> g5_;
  const ParamGenerator<T6> g6_;
  const ParamGenerator<T7> g7_;
  const ParamGenerator<T8> g8_;
  const ParamGenerator<T9> g9_;
};  // class CartesianProductGenerator9


template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7, typename T8, typename T9, typename T10>
class CartesianProductGenerator10
    : public ParamGeneratorInterface< ::testing::tuple<T1, T2, T3, T4, T5, T6,
        T7, T8, T9, T10> > {
 public:
  typedef ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10> ParamType;

  CartesianProductGenerator10(const ParamGenerator<T1>& g1,
      const ParamGenerator<T2>& g2, const ParamGenerator<T3>& g3,
      const ParamGenerator<T4>& g4, const ParamGenerator<T5>& g5,
      const ParamGenerator<T6>& g6, const ParamGenerator<T7>& g7,
      const ParamGenerator<T8>& g8, const ParamGenerator<T9>& g9,
      const ParamGenerator<T10>& g10)
      : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7), g8_(g8),
          g9_(g9), g10_(g10) {}
  virtual ~CartesianProductGenerator10() {}

  virtual ParamIteratorInterface<ParamType>* Begin() const {
    return new Iterator(this, g1_, g1_.begin(), g2_, g2_.begin(), g3_,
        g3_.begin(), g4_, g4_.begin(), g5_, g5_.begin(), g6_, g6_.begin(), g7_,
        g7_.begin(), g8_, g8_.begin(), g9_, g9_.begin(), g10_, g10_.begin());
  }
  virtual ParamIteratorInterface<ParamType>* End() const {
    return new Iterator(this, g1_, g1_.end(), g2_, g2_.end(), g3_, g3_.end(),
        g4_, g4_.end(), g5_, g5_.end(), g6_, g6_.end(), g7_, g7_.end(), g8_,
        g8_.end(), g9_, g9_.end(), g10_, g10_.end());
  }

 private:
  class Iterator : public ParamIteratorInterface<ParamType> {
   public:
    Iterator(const ParamGeneratorInterface<ParamType>* base,
      const ParamGenerator<T1>& g1,
      const typename ParamGenerator<T1>::iterator& current1,
      const ParamGenerator<T2>& g2,
      const typename ParamGenerator<T2>::iterator& current2,
      const ParamGenerator<T3>& g3,
      const typename ParamGenerator<T3>::iterator& current3,
      const ParamGenerator<T4>& g4,
      const typename ParamGenerator<T4>::iterator& current4,
      const ParamGenerator<T5>& g5,
      const typename ParamGenerator<T5>::iterator& current5,
      const ParamGenerator<T6>& g6,
      const typename ParamGenerator<T6>::iterator& current6,
      const ParamGenerator<T7>& g7,
      const typename ParamGenerator<T7>::iterator& current7,
      const ParamGenerator<T8>& g8,
      const typename ParamGenerator<T8>::iterator& current8,
      const ParamGenerator<T9>& g9,
      const typename ParamGenerator<T9>::iterator& current9,
      const ParamGenerator<T10>& g10,
      const typename ParamGenerator<T10>::iterator& current10)
        : base_(base),
          begin1_(g1.begin()), end1_(g1.end()), current1_(current1),
          begin2_(g2.begin()), end2_(g2.end()), current2_(current2),
          begin3_(g3.begin()), end3_(g3.end()), current3_(current3),
          begin4_(g4.begin()), end4_(g4.end()), current4_(current4),
          begin5_(g5.begin()), end5_(g5.end()), current5_(current5),
          begin6_(g6.begin()), end6_(g6.end()), current6_(current6),
          begin7_(g7.begin()), end7_(g7.end()), current7_(current7),
          begin8_(g8.begin()), end8_(g8.end()), current8_(current8),
          begin9_(g9.begin()), end9_(g9.end()), current9_(current9),
          begin10_(g10.begin()), end10_(g10.end()), current10_(current10)    {
      ComputeCurrentValue();
    }
    virtual ~Iterator() {}

    virtual const ParamGeneratorInterface<ParamType>* BaseGenerator() const {
      return base_;
    }
    // Advance should not be called on beyond-of-range iterators
    // so no component iterators must be beyond end of range, either.
    virtual void Advance() {
      assert(!AtEnd());
      ++current10_;
      if (current10_ == end10_) {
        current10_ = begin10_;
        ++current9_;
      }
      if (current9_ == end9_) {
        current9_ = begin9_;
        ++current8_;
      }
      if (current8_ == end8_) {
        current8_ = begin8_;
        ++current7_;
      }
      if (current7_ == end7_) {
        current7_ = begin7_;
        ++current6_;
      }
      if (current6_ == end6_) {
        current6_ = begin6_;
        ++current5_;
      }
      if (current5_ == end5_) {
        current5_ = begin5_;
        ++current4_;
      }
      if (current4_ == end4_) {
        current4_ = begin4_;
        ++current3_;
      }
      if (current3_ == end3_) {
        current3_ = begin3_;
        ++current2_;
      }
      if (current2_ == end2_) {
        current2_ = begin2_;
        ++current1_;
      }
      ComputeCurrentValue();
    }
    virtual ParamIteratorInterface<ParamType>* Clone() const {
      return new Iterator(*this);
    }
    virtual const ParamType* Current() const { return current_value_.get(); }
    virtual bool Equals(const ParamIteratorInterface<ParamType>& other) const {
      // Having the same base generator guarantees that the other
      // iterator is of the same type and we can downcast.
      GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
          « "The program attempted to compare iterators "
          « "from different generators." « std::endl;
      const Iterator* typed_other =
          CheckedDowncastToActualType<const Iterator>(&other);
      // We must report iterators equal if they both point beyond their
      // respective ranges. That can happen in a variety of fashions,
      // so we have to consult AtEnd().
      return (AtEnd() && typed_other->AtEnd()) ||
         (
          current1_ == typed_other->current1_ &&
          current2_ == typed_other->current2_ &&
          current3_ == typed_other->current3_ &&
          current4_ == typed_other->current4_ &&
          current5_ == typed_other->current5_ &&
          current6_ == typed_other->current6_ &&
          current7_ == typed_other->current7_ &&
          current8_ == typed_other->current8_ &&
          current9_ == typed_other->current9_ &&
          current10_ == typed_other->current10_);
    }

   private:
    Iterator(const Iterator& other)
        : base_(other.base_),
        begin1_(other.begin1_),
        end1_(other.end1_),
        current1_(other.current1_),
        begin2_(other.begin2_),
        end2_(other.end2_),
        current2_(other.current2_),
        begin3_(other.begin3_),
        end3_(other.end3_),
        current3_(other.current3_),
        begin4_(other.begin4_),
        end4_(other.end4_),
        current4_(other.current4_),
        begin5_(other.begin5_),
        end5_(other.end5_),
        current5_(other.current5_),
        begin6_(other.begin6_),
        end6_(other.end6_),
        current6_(other.current6_),
        begin7_(other.begin7_),
        end7_(other.end7_),
        current7_(other.current7_),
        begin8_(other.begin8_),
        end8_(other.end8_),
        current8_(other.current8_),
        begin9_(other.begin9_),
        end9_(other.end9_),
        current9_(other.current9_),
        begin10_(other.begin10_),
        end10_(other.end10_),
        current10_(other.current10_) {
      ComputeCurrentValue();
    }

    void ComputeCurrentValue() {
      if (!AtEnd())
        current_value_.reset(new ParamType(*current1_, *current2_, *current3_,
            *current4_, *current5_, *current6_, *current7_, *current8_,
            *current9_, *current10_));
    }
    bool AtEnd() const {
      // We must report iterator past the end of the range when either of the
      // component iterators has reached the end of its range.
      return
          current1_ == end1_ ||
          current2_ == end2_ ||
          current3_ == end3_ ||
          current4_ == end4_ ||
          current5_ == end5_ ||
          current6_ == end6_ ||
          current7_ == end7_ ||
          current8_ == end8_ ||
          current9_ == end9_ ||
          current10_ == end10_;
    }

    // No implementation - assignment is unsupported.
    void operator=(const Iterator& other);

    const ParamGeneratorInterface<ParamType>* const base_;
    // begin[i]_ and end[i]_ define the i-th range that Iterator traverses.
    // current[i]_ is the actual traversing iterator.
    const typename ParamGenerator<T1>::iterator begin1_;
    const typename ParamGenerator<T1>::iterator end1_;
    typename ParamGenerator<T1>::iterator current1_;
    const typename ParamGenerator<T2>::iterator begin2_;
    const typename ParamGenerator<T2>::iterator end2_;
```

```
    typename ParamGenerator<T2>::iterator current2_;
    const typename ParamGenerator<T3>::iterator begin3_;
    const typename ParamGenerator<T3>::iterator end3_;
    typename ParamGenerator<T3>::iterator current3_;
    const typename ParamGenerator<T4>::iterator begin4_;
    const typename ParamGenerator<T4>::iterator end4_;
    typename ParamGenerator<T4>::iterator current4_;
    const typename ParamGenerator<T5>::iterator begin5_;
    const typename ParamGenerator<T5>::iterator end5_;
    typename ParamGenerator<T5>::iterator current5_;
    const typename ParamGenerator<T6>::iterator begin6_;
    const typename ParamGenerator<T6>::iterator end6_;
    typename ParamGenerator<T6>::iterator current6_;
    const typename ParamGenerator<T7>::iterator begin7_;
    const typename ParamGenerator<T7>::iterator end7_;
    typename ParamGenerator<T7>::iterator current7_;
    const typename ParamGenerator<T8>::iterator begin8_;
    const typename ParamGenerator<T8>::iterator end8_;
    typename ParamGenerator<T8>::iterator current8_;
    const typename ParamGenerator<T9>::iterator begin9_;
    const typename ParamGenerator<T9>::iterator end9_;
    typename ParamGenerator<T9>::iterator current9_;
    const typename ParamGenerator<T10>::iterator begin10_;
    const typename ParamGenerator<T10>::iterator end10_;
    typename ParamGenerator<T10>::iterator current10_;
    linked_ptr<ParamType> current_value_;
  };  // class CartesianProductGenerator10::Iterator

  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductGenerator10& other);

  const ParamGenerator<T1> g1_;
  const ParamGenerator<T2> g2_;
  const ParamGenerator<T3> g3_;
  const ParamGenerator<T4> g4_;
  const ParamGenerator<T5> g5_;
  const ParamGenerator<T6> g6_;
  const ParamGenerator<T7> g7_;
  const ParamGenerator<T8> g8_;
  const ParamGenerator<T9> g9_;
  const ParamGenerator<T10> g10_;
};  // class CartesianProductGenerator10


// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
//
// Helper classes providing Combine() with polymorphic features. They allow
// casting CartesianProductGeneratorN<T> to ParamGenerator<U> if T is
// convertible to U.
//
template <class Generator1, class Generator2>
class CartesianProductHolder2 {
 public:
CartesianProductHolder2(const Generator1& g1, const Generator2& g2)
      : g1_(g1), g2_(g2) {}
  template <typename T1, typename T2>
  operator ParamGenerator< ::testing::tuple<T1, T2> >() const {
    return ParamGenerator< ::testing::tuple<T1, T2> >(
        new CartesianProductGenerator2<T1, T2>(
        static_cast<ParamGenerator<T1> >(g1_),
        static_cast<ParamGenerator<T2> >(g2_)));
  }

 private:
  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductHolder2& other);

  const Generator1 g1_;
  const Generator2 g2_;
};  // class CartesianProductHolder2

template <class Generator1, class Generator2, class Generator3>
class CartesianProductHolder3 {
 public:
CartesianProductHolder3(const Generator1& g1, const Generator2& g2,
    const Generator3& g3)
      : g1_(g1), g2_(g2), g3_(g3) {}
  template <typename T1, typename T2, typename T3>
  operator ParamGenerator< ::testing::tuple<T1, T2, T3> >() const {
    return ParamGenerator< ::testing::tuple<T1, T2, T3> >(
        new CartesianProductGenerator3<T1, T2, T3>(
        static_cast<ParamGenerator<T1> >(g1_),
        static_cast<ParamGenerator<T2> >(g2_),
        static_cast<ParamGenerator<T3> >(g3_)));
  }

 private:
  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductHolder3& other);

  const Generator1 g1_;
  const Generator2 g2_;
  const Generator3 g3_;
};  // class CartesianProductHolder3

template <class Generator1, class Generator2, class Generator3,
    class Generator4>
class CartesianProductHolder4 {
 public:
CartesianProductHolder4(const Generator1& g1, const Generator2& g2,
    const Generator3& g3, const Generator4& g4)
      : g1_(g1), g2_(g2), g3_(g3), g4_(g4) {}
  template <typename T1, typename T2, typename T3, typename T4>
  operator ParamGenerator< ::testing::tuple<T1, T2, T3, T4> >() const {
    return ParamGenerator< ::testing::tuple<T1, T2, T3, T4> >(
        new CartesianProductGenerator4<T1, T2, T3, T4>(
        static_cast<ParamGenerator<T1> >(g1_),
        static_cast<ParamGenerator<T2> >(g2_),
        static_cast<ParamGenerator<T3> >(g3_),
        static_cast<ParamGenerator<T4> >(g4_)));
  }

 private:
  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductHolder4& other);

  const Generator1 g1_;
  const Generator2 g2_;
  const Generator3 g3_;
  const Generator4 g4_;
};  // class CartesianProductHolder4

template <class Generator1, class Generator2, class Generator3,
    class Generator4, class Generator5>
class CartesianProductHolder5 {
 public:
CartesianProductHolder5(const Generator1& g1, const Generator2& g2,
    const Generator3& g3, const Generator4& g4, const Generator5& g5)
      : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5) {}
  template <typename T1, typename T2, typename T3, typename T4, typename T5>
  operator ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5> >() const {
    return ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5> >(
        new CartesianProductGenerator5<T1, T2, T3, T4, T5>(
        static_cast<ParamGenerator<T1> >(g1_),
        static_cast<ParamGenerator<T2> >(g2_),
        static_cast<ParamGenerator<T3> >(g3_),
        static_cast<ParamGenerator<T4> >(g4_),
        static_cast<ParamGenerator<T5> >(g5_)));
  }

 private:
  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductHolder5& other);

  const Generator1 g1_;
  const Generator2 g2_;
  const Generator3 g3_;
  const Generator4 g4_;
  const Generator5 g5_;
};  // class CartesianProductHolder5

template <class Generator1, class Generator2, class Generator3,
    class Generator4, class Generator5, class Generator6>
class CartesianProductHolder6 {
 public:
CartesianProductHolder6(const Generator1& g1, const Generator2& g2,
    const Generator3& g3, const Generator4& g4, const Generator5& g5,
    const Generator6& g6)
      : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6) {}
  template <typename T1, typename T2, typename T3, typename T4, typename T5,
      typename T6>
  operator ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6> >() const {
    return ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6> >(
        new CartesianProductGenerator6<T1, T2, T3, T4, T5, T6>(
        static_cast<ParamGenerator<T1> >(g1_),
        static_cast<ParamGenerator<T2> >(g2_),
        static_cast<ParamGenerator<T3> >(g3_),
        static_cast<ParamGenerator<T4> >(g4_),
        static_cast<ParamGenerator<T5> >(g5_),
        static_cast<ParamGenerator<T6> >(g6_)));
  }

 private:
  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductHolder6& other);

  const Generator1 g1_;
  const Generator2 g2_;
  const Generator3 g3_;
  const Generator4 g4_;
  const Generator5 g5_;
  const Generator6 g6_;
};  // class CartesianProductHolder6

template <class Generator1, class Generator2, class Generator3,
    class Generator4, class Generator5, class Generator6, class Generator7>
class CartesianProductHolder7 {
 public:
CartesianProductHolder7(const Generator1& g1, const Generator2& g2,
    const Generator3& g3, const Generator4& g4, const Generator5& g5,
    const Generator6& g6, const Generator7& g7)
      : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7) {}
  template <typename T1, typename T2, typename T3, typename T4, typename T5,
      typename T6, typename T7>
  operator ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6,
      T7> >() const {
    return ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6, T7> >(
        new CartesianProductGenerator7<T1, T2, T3, T4, T5, T6, T7>(
        static_cast<ParamGenerator<T1> >(g1_),
        static_cast<ParamGenerator<T2> >(g2_),
        static_cast<ParamGenerator<T3> >(g3_),
        static_cast<ParamGenerator<T4> >(g4_),
        static_cast<ParamGenerator<T5> >(g5_),
        static_cast<ParamGenerator<T6> >(g6_),
        static_cast<ParamGenerator<T7> >(g7_)));
  }

 private:
  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductHolder7& other);

  const Generator1 g1_;
  const Generator2 g2_;
  const Generator3 g3_;
  const Generator4 g4_;
  const Generator5 g5_;
  const Generator6 g6_;
  const Generator7 g7_;
};  // class CartesianProductHolder7

template <class Generator1, class Generator2, class Generator3,
    class Generator4, class Generator5, class Generator6, class Generator7,
    class Generator8>
class CartesianProductHolder8 {
 public:
CartesianProductHolder8(const Generator1& g1, const Generator2& g2,
    const Generator3& g3, const Generator4& g4, const Generator5& g5,
    const Generator6& g6, const Generator7& g7, const Generator8& g8)
      : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7),
          g8_(g8) {}
  template <typename T1, typename T2, typename T3, typename T4, typename T5,
      typename T6, typename T7, typename T8>
  operator ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6, T7,
      T8> >() const {
    return ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8> >(
        new CartesianProductGenerator8<T1, T2, T3, T4, T5, T6, T7, T8>(
        static_cast<ParamGenerator<T1> >(g1_),
        static_cast<ParamGenerator<T2> >(g2_),
        static_cast<ParamGenerator<T3> >(g3_),
        static_cast<ParamGenerator<T4> >(g4_),
        static_cast<ParamGenerator<T5> >(g5_),
        static_cast<ParamGenerator<T6> >(g6_),
        static_cast<ParamGenerator<T7> >(g7_),
        static_cast<ParamGenerator<T8> >(g8_)));
  }

 private:
  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductHolder8& other);

  const Generator1 g1_;
  const Generator2 g2_;
  const Generator3 g3_;
  const Generator4 g4_;
  const Generator5 g5_;
  const Generator6 g6_;
  const Generator7 g7_;
  const Generator8 g8_;
};  // class CartesianProductHolder8

template <class Generator1, class Generator2, class Generator3,
    class Generator4, class Generator5, class Generator6, class Generator7,
    class Generator8, class Generator9>
class CartesianProductHolder9 {
 public:
CartesianProductHolder9(const Generator1& g1, const Generator2& g2,
    const Generator3& g3, const Generator4& g4, const Generator5& g5,
    const Generator6& g6, const Generator7& g7, const Generator8& g8,
    const Generator9& g9)
      : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7), g8_(g8),
          g9_(g9) {}
  template <typename T1, typename T2, typename T3, typename T4, typename T5,
      typename T6, typename T7, typename T8, typename T9>
  operator ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8,
      T9> >() const {
    return ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8,
        T9> >(
        new CartesianProductGenerator9<T1, T2, T3, T4, T5, T6, T7, T8, T9>(
        static_cast<ParamGenerator<T1> >(g1_),
        static_cast<ParamGenerator<T2> >(g2_),
        static_cast<ParamGenerator<T3> >(g3_),
        static_cast<ParamGenerator<T4> >(g4_),
        static_cast<ParamGenerator<T5> >(g5_),
        static_cast<ParamGenerator<T6> >(g6_),
        static_cast<ParamGenerator<T7> >(g7_),
        static_cast<ParamGenerator<T8> >(g8_),
        static_cast<ParamGenerator<T9> >(g9_)));
  }

 private:
  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductHolder9& other);

  const Generator1 g1_;
  const Generator2 g2_;
  const Generator3 g3_;
  const Generator4 g4_;
  const Generator5 g5_;
  const Generator6 g6_;
  const Generator7 g7_;
  const Generator8 g8_;
  const Generator9 g9_;
};  // class CartesianProductHolder9

template <class Generator1, class Generator2, class Generator3,
    class Generator4, class Generator5, class Generator6, class Generator7,
    class Generator8, class Generator9, class Generator10>
class CartesianProductHolder10 {
 public:
CartesianProductHolder10(const Generator1& g1, const Generator2& g2,
    const Generator3& g3, const Generator4& g4, const Generator5& g5,
    const Generator6& g6, const Generator7& g7, const Generator8& g8,
    const Generator9& g9, const Generator10& g10)
      : g1_(g1), g2_(g2), g3_(g3), g4_(g4), g5_(g5), g6_(g6), g7_(g7), g8_(g8),
          g9_(g9), g10_(g10) {}
  template <typename T1, typename T2, typename T3, typename T4, typename T5,
      typename T6, typename T7, typename T8, typename T9, typename T10>
  operator ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8, T9,
      T10> >() const {
    return ParamGenerator< ::testing::tuple<T1, T2, T3, T4, T5, T6, T7, T8, T9,
        T10> >(
        new CartesianProductGenerator10<T1, T2, T3, T4, T5, T6, T7, T8, T9,
            T10>(
        static_cast<ParamGenerator<T1> >(g1_),
        static_cast<ParamGenerator<T2> >(g2_),
        static_cast<ParamGenerator<T3> >(g3_),
        static_cast<ParamGenerator<T4> >(g4_),
        static_cast<ParamGenerator<T5> >(g5_),
        static_cast<ParamGenerator<T6> >(g6_),
        static_cast<ParamGenerator<T7> >(g7_),
        static_cast<ParamGenerator<T8> >(g8_),
        static_cast<ParamGenerator<T9> >(g9_),
        static_cast<ParamGenerator<T10> >(g10_)));
  }

 private:
  // No implementation - assignment is unsupported.
  void operator=(const CartesianProductHolder10& other);

  const Generator1 g1_;
  const Generator2 g2_;
  const Generator3 g3_;
  const Generator4 g4_;
  const Generator5 g5_;
  const Generator6 g6_;
  const Generator7 g7_;
  const Generator8 g8_;
  const Generator9 g9_;
  const Generator10 g10_;
};  // class CartesianProductHolder10

# endif  // GTEST_HAS_COMBINE

}  // namespace internal
}  // namespace testing

#endif  // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PARAM_UTIL_GENERATED_H_
```

9.43 Dokumentacja pliku packages/Microsoft.googletest.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.

```
// Copyright 2008 Google Inc.
// All Rights Reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


// Type and function utilities for implementing parameterized tests.

// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PARAM_UTIL_H_
#define GTEST_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"

namespace testing {

// Input to a parameterized test name generator, describing a test parameter.
// Consists of the parameter value and the integer parameter index.
template <class ParamType>
struct TestParamInfo {
  TestParamInfo(const ParamType& a_param, size_t an_index) :
    param(a_param),
    index(an_index) {}
  ParamType param;
  size_t index;
};

```

```
// A builtin parameterized test name generator which returns the result of
// testing::PrintToString.
struct PrintToStringParamName {
  template <class ParamType>
  std::string operator()(const TestParamInfo<ParamType>& info) const {
    return PrintToString(info.param);
  }
};

namespace internal {

// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
//
// Outputs a message explaining invalid registration of different
// fixture class for the same test case. This may happen when
// TEST_P macro is used to define two tests with the same name
// but in different namespaces.
GTEST_API_ void ReportInvalidTestCaseType(const char* test_case_name,
                                          CodeLocation code_location);

template <typename> class ParamGeneratorInterface;
template <typename> class ParamGenerator;

// Interface for iterating over elements provided by an implementation
// of ParamGeneratorInterface<T>.
template <typename T>
class ParamIteratorInterface {
 public:
  virtual ~ParamIteratorInterface() {}
  // A pointer to the base generator instance.
  // Used only for the purposes of iterator comparison
  // to make sure that two iterators belong to the same generator.
  virtual const ParamGeneratorInterface<T>* BaseGenerator() const = 0;
  // Advances iterator to point to the next element
  // provided by the generator. The caller is responsible
  // for not calling Advance() on an iterator equal to
  // BaseGenerator()->End().
  virtual void Advance() = 0;
  // Clones the iterator object. Used for implementing copy semantics
  // of ParamIterator<T>.
  virtual ParamIteratorInterface* Clone() const = 0;
  // Dereferences the current iterator and provides (read-only) access
  // to the pointed value. It is the caller's responsibility not to call
  // Current() on an iterator equal to BaseGenerator()->End().
  // Used for implementing ParamGenerator<T>::operator*().
  virtual const T* Current() const = 0;
  // Determines whether the given iterator and other point to the same
  // element in the sequence generated by the generator.
  // Used for implementing ParamGenerator<T>::operator==().
  virtual bool Equals(const ParamIteratorInterface& other) const = 0;
};

// Class iterating over elements provided by an implementation of
// ParamGeneratorInterface<T>. It wraps ParamIteratorInterface<T>
// and implements the const forward iterator concept.
template <typename T>
class ParamIterator {
 public:
  typedef T value_type;
  typedef const T& reference;
  typedef ptrdiff_t difference_type;

  // ParamIterator assumes ownership of the impl_ pointer.
  ParamIterator(const ParamIterator& other) : impl_(other.impl_->Clone()) {}
  ParamIterator& operator=(const ParamIterator& other) {
    if (this != &other)
      impl_.reset(other.impl_->Clone());
    return *this;
  }

  const T& operator*() const { return *impl_->Current(); }
  const T* operator->() const { return impl_->Current(); }
  // Prefix version of operator++.
  ParamIterator& operator++() {
    impl_->Advance();
    return *this;
  }
  // Postfix version of operator++.
  ParamIterator operator++(int /*unused*/) {
    ParamIteratorInterface<T>* clone = impl_->Clone();
    impl_->Advance();
    return ParamIterator(clone);
  }
  bool operator==(const ParamIterator& other) const {
    return impl_.get() == other.impl_.get() || impl_->Equals(*other.impl_);
  }
  bool operator!=(const ParamIterator& other) const {
    return !(*this == other);
  }

 private:
  friend class ParamGenerator<T>;
  explicit ParamIterator(ParamIteratorInterface<T>* impl) : impl_(impl) {}
  scoped_ptr<ParamIteratorInterface<T> > impl_;
};

// ParamGeneratorInterface<T> is the binary interface to access generators
// defined in other translation units.
template <typename T>
class ParamGeneratorInterface {
 public:
  typedef T ParamType;

  virtual ~ParamGeneratorInterface() {}

  // Generator interface definition
  virtual ParamIteratorInterface<T>* Begin() const = 0;
  virtual ParamIteratorInterface<T>* End() const = 0;
};

// Wraps ParamGeneratorInterface<T> and provides general generator syntax
// compatible with the STL Container concept.
// This class implements copy initialization semantics and the contained
// ParamGeneratorInterface<T> instance is shared among all copies
// of the original object. This is possible because that instance is immutable.
template<typename T>
class ParamGenerator {
 public:
  typedef ParamIterator<T> iterator;

  explicit ParamGenerator(ParamGeneratorInterface<T>* impl) : impl_(impl) {}
  ParamGenerator(const ParamGenerator& other) : impl_(other.impl_) {}

  ParamGenerator& operator=(const ParamGenerator& other) {
    impl_ = other.impl_;
    return *this;
  }

  iterator begin() const { return iterator(impl_->Begin()); }
  iterator end() const { return iterator(impl_->End()); }

 private:
  linked_ptr<const ParamGeneratorInterface<T> > impl_;
};

// Generates values from a range of two comparable values. Can be used to
// generate sequences of user-defined types that implement operator+() and
// operator<().
// This class is used in the Range() function.
template <typename T, typename IncrementT>
class RangeGenerator : public ParamGeneratorInterface<T> {
 public:
  RangeGenerator(T begin, T end, IncrementT step)
      : begin_(begin), end_(end),
        step_(step), end_index_(CalculateEndIndex(begin, end, step)) {}
  virtual ~RangeGenerator() {}

  virtual ParamIteratorInterface<T>* Begin() const {
    return new Iterator(this, begin_, 0, step_);
  }
  virtual ParamIteratorInterface<T>* End() const {
    return new Iterator(this, end_, end_index_, step_);
  }

 private:
  class Iterator : public ParamIteratorInterface<T> {
   public:
    Iterator(const ParamGeneratorInterface<T>* base, T value, int index,
             IncrementT step)
        : base_(base), value_(value), index_(index), step_(step) {}
    virtual ~Iterator() {}

    virtual const ParamGeneratorInterface<T>* BaseGenerator() const {
      return base_;
    }
    virtual void Advance() {
      value_ = static_cast<T>(value_ + step_);
      index_++;
    }
    virtual ParamIteratorInterface<T>* Clone() const {
      return new Iterator(*this);
    }
    virtual const T* Current() const { return &value_; }
    virtual bool Equals(const ParamIteratorInterface<T>& other) const {
      // Having the same base generator guarantees that the other
      // iterator is of the same type and we can downcast.
      GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
          « "The program attempted to compare iterators "
          « "from different generators." « std::endl;
      const int other_index =
          CheckedDowncastToActualType<const Iterator>(&other)->index_;
      return index_ == other_index;
    }

   private:
    Iterator(const Iterator& other)
        : ParamIteratorInterface<T>(),
          base_(other.base_), value_(other.value_), index_(other.index_),
          step_(other.step_) {}

    // No implementation - assignment is unsupported.
    void operator=(const Iterator& other);

    const ParamGeneratorInterface<T>* const base_;
    T value_;
    int index_;
    const IncrementT step_;
  };  // class RangeGenerator::Iterator

  static int CalculateEndIndex(const T& begin,
                               const T& end,
                               const IncrementT& step) {
    int end_index = 0;
    for (T i = begin; i < end; i = static_cast<T>(i + step))
      end_index++;
    return end_index;
  }

  // No implementation - assignment is unsupported.
  void operator=(const RangeGenerator& other);

  const T begin_;
  const T end_;
  const IncrementT step_;
  // The index for the end() iterator. All the elements in the generated
  // sequence are indexed (0-based) to aid iterator comparison.
  const int end_index_;
};  // class RangeGenerator


// Generates values from a pair of STL-style iterators. Used in the
// ValuesIn() function. The elements are copied from the source range
// since the source can be located on the stack, and the generator
// is likely to persist beyond that stack frame.
template <typename T>
class ValuesInIteratorRangeGenerator : public ParamGeneratorInterface<T> {
 public:
  template <typename ForwardIterator>
  ValuesInIteratorRangeGenerator(ForwardIterator begin, ForwardIterator end)
      : container_(begin, end) {}
  virtual ~ValuesInIteratorRangeGenerator() {}

  virtual ParamIteratorInterface<T>* Begin() const {
    return new Iterator(this, container_.begin());
  }
  virtual ParamIteratorInterface<T>* End() const {
    return new Iterator(this, container_.end());
  }

 private:
  typedef typename ::std::vector<T> ContainerType;

  class Iterator : public ParamIteratorInterface<T> {
   public:
    Iterator(const ParamGeneratorInterface<T>* base,
             typename ContainerType::const_iterator iterator)
        : base_(base), iterator_(iterator) {}
    virtual ~Iterator() {}

    virtual const ParamGeneratorInterface<T>* BaseGenerator() const {
      return base_;
    }
    virtual void Advance() {
      ++iterator_;
      value_.reset();
    }
    virtual ParamIteratorInterface<T>* Clone() const {
      return new Iterator(*this);
    }
    // We need to use cached value referenced by iterator_ because *iterator_
    // can return a temporary object (and of type other then T), so just
    // having "return &*iterator_;" doesn't work.
    // value_ is updated here and not in Advance() because Advance()
    // can advance iterator_ beyond the end of the range, and we cannot
    // detect that fact. The client code, on the other hand, is
    // responsible for not calling Current() on an out-of-range iterator.
    virtual const T* Current() const {
      if (value_.get() == NULL)
        value_.reset(new T(*iterator_));
      return value_.get();
    }
    virtual bool Equals(const ParamIteratorInterface<T>& other) const {
      // Having the same base generator guarantees that the other
      // iterator is of the same type and we can downcast.
      GTEST_CHECK_(BaseGenerator() == other.BaseGenerator())
          « "The program attempted to compare iterators "
          « "from different generators." « std::endl;
      return iterator_ ==
          CheckedDowncastToActualType<const Iterator>(&other)->iterator_;
    }

   private:
    Iterator(const Iterator& other)
          // The explicit constructor call suppresses a false warning
          // emitted by gcc when supplied with the -Wextra option.
        : ParamIteratorInterface<T>(),
          base_(other.base_),
          iterator_(other.iterator_) {}

    const ParamGeneratorInterface<T>* const base_;
    typename ContainerType::const_iterator iterator_;
    // A cached value of *iterator_. We keep it here to allow access by
    // pointer in the wrapping iterator's operator->().
    // value_ needs to be mutable to be accessed in Current().
    // Use of scoped_ptr helps manage cached value's lifetime,
    // which is bound by the lifespan of the iterator itself.
    mutable scoped_ptr<const T> value_;
  };  // class ValuesInIteratorRangeGenerator::Iterator

  // No implementation - assignment is unsupported.
  void operator=(const ValuesInIteratorRangeGenerator& other);

  const ContainerType container_;
};  // class ValuesInIteratorRangeGenerator

// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
//
// Default parameterized test name generator, returns a string containing the
// integer test parameter index.
template <class ParamType>
std::string DefaultParamName(const TestParamInfo<ParamType>& info) {
  Message name_stream;
  name_stream « info.index;
  return name_stream.GetString();
}

// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
//
// Parameterized test name overload helpers, which help the
// INSTANTIATE_TEST_CASE_P macro choose between the default parameterized
// test name generator and user param name generator.
template <class ParamType, class ParamNameGenFunctor>
ParamNameGenFunctor GetParamNameGen(ParamNameGenFunctor func) {
  return func;
}

template <class ParamType>
struct ParamNameGenFunc {
  typedef std::string Type(const TestParamInfo<ParamType>&);
};

template <class ParamType>
typename ParamNameGenFunc<ParamType>::Type *GetParamNameGen() {
  return DefaultParamName;
}

// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
//
// Stores a parameter value and later creates tests parameterized with that
// value.
template <class TestClass>
class ParameterizedTestFactory : public TestFactoryBase {
 public:
  typedef typename TestClass::ParamType ParamType;
  explicit ParameterizedTestFactory(ParamType parameter) :
      parameter_(parameter) {}
  virtual Test* CreateTest() {
    TestClass::SetParam(&parameter_);
    return new TestClass();
  }

 private:
  const ParamType parameter_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(ParameterizedTestFactory);
};

// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
//
// TestMetaFactoryBase is a base class for meta-factories that create
// test factories for passing into MakeAndRegisterTestInfo function.
template <class ParamType>
class TestMetaFactoryBase {
 public:
  virtual ~TestMetaFactoryBase() {}

  virtual TestFactoryBase* CreateTestFactory(ParamType parameter) = 0;
};

// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
//
// TestMetaFactory creates test factories for passing into
// MakeAndRegisterTestInfo function. Since MakeAndRegisterTestInfo receives
// ownership of test factory pointer, same factory object cannot be passed
// into that method twice. But ParameterizedTestCaseInfo is going to call
// it for each Test/Parameter value combination. Thus it needs meta factory
// creator class.
template <class TestCase>
class TestMetaFactory
    : public TestMetaFactoryBase<typename TestCase::ParamType> {
 public:
  typedef typename TestCase::ParamType ParamType;

  TestMetaFactory() {}

  virtual TestFactoryBase* CreateTestFactory(ParamType parameter) {
    return new ParameterizedTestFactory<TestCase>(parameter);
  }

 private:
  GTEST_DISALLOW_COPY_AND_ASSIGN_(TestMetaFactory);
};

// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
//
// ParameterizedTestCaseInfoBase is a generic interface
// to ParameterizedTestCaseInfo classes. ParameterizedTestCaseInfoBase
// accumulates test information provided by TEST_P macro invocations
// and generators provided by INSTANTIATE_TEST_CASE_P macro invocations
// and uses that information to register all resulting test instances
// in RegisterTests method. The ParameterizeTestCaseRegistry class holds
// a collection of pointers to the ParameterizedTestCaseInfo objects
// and calls RegisterTests() on each of them when asked.
class ParameterizedTestCaseInfoBase {
 public:
  virtual ~ParameterizedTestCaseInfoBase() {}

  // Base part of test case name for display purposes.
  virtual const std::string& GetTestCaseName() const = 0;
  // Test case id to verify identity.
  virtual TypeId GetTestCaseTypeId() const = 0;
  // UnitTest class invokes this method to register tests in this
  // test case right before running them in RUN_ALL_TESTS macro.
  // This method should not be called more then once on any single
  // instance of a ParameterizedTestCaseInfoBase derived class.
  virtual void RegisterTests() = 0;

 protected:
  ParameterizedTestCaseInfoBase() {}

 private:
  GTEST_DISALLOW_COPY_AND_ASSIGN_(ParameterizedTestCaseInfoBase);
};

// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
//
// ParameterizedTestCaseInfo accumulates tests obtained from TEST_P
// macro invocations for a particular test case and generators
// obtained from INSTANTIATE_TEST_CASE_P macro invocations for that
// test case. It registers tests with all values generated by all
// generators when asked.
template <class TestCase>
class ParameterizedTestCaseInfo : public ParameterizedTestCaseInfoBase {
 public:
  // ParamType and GeneratorCreationFunc are private types but are required
  // for declarations of public methods AddTestPattern() and
  // AddTestCaseInstantiation().
  typedef typename TestCase::ParamType ParamType;
  // A function that returns an instance of appropriate generator type.
  typedef ParamGenerator<ParamType>(GeneratorCreationFunc)();
  typedef typename ParamNameGenFunc<ParamType>::Type ParamNameGeneratorFunc;

  explicit ParameterizedTestCaseInfo(
      const char* name, CodeLocation code_location)
      : test_case_name_(name), code_location_(code_location) {}

  // Test case base name for display purposes.
  virtual const std::string& GetTestCaseName() const { return test_case_name_; }
  // Test case id to verify identity.
  virtual TypeId GetTestCaseTypeId() const { return GetTypeId<TestCase>(); }
  // TEST_P macro uses AddTestPattern() to record information
  // about a single test in a LocalTestInfo structure.
  // test_case_name is the base name of the test case (without invocation
  // prefix). test_base_name is the name of an individual test without
  // parameter index. For the test SequenceA/FooTest.DoBar/1 FooTest is
  // test case base name and DoBar is test base name.
  void AddTestPattern(const char* test_case_name,
                      const char* test_base_name,
                      TestMetaFactoryBase<ParamType>* meta_factory) {
    tests_.push_back(linked_ptr<TestInfo>(new TestInfo(test_case_name,
                                                       test_base_name,
                                                       meta_factory)));
  }
  // INSTANTIATE_TEST_CASE_P macro uses AddGenerator() to record information
  // about a generator.
  int AddTestCaseInstantiation(const std::string& instantiation_name,
                               GeneratorCreationFunc* func,
                               ParamNameGeneratorFunc* name_func,
                               const char* file, int line) {
    instantiations_.push_back(
        InstantiationInfo(instantiation_name, func, name_func, file, line));
    return 0;  // Return value used only to run this method in namespace scope.
  }
  // UnitTest class invokes this method to register tests in this test case
  // test cases right before running tests in RUN_ALL_TESTS macro.
  // This method should not be called more then once on any single
  // instance of a ParameterizedTestCaseInfoBase derived class.
  // UnitTest has a guard to prevent from calling this method more then once.
  virtual void RegisterTests() {
    for (typename TestInfoContainer::iterator test_it = tests_.begin();
         test_it != tests_.end(); ++test_it) {
      linked_ptr<TestInfo> test_info = *test_it;
      for (typename InstantiationContainer::iterator gen_it =
               instantiations_.begin(); gen_it != instantiations_.end();
               ++gen_it) {
        const std::string& instantiation_name = gen_it->name;
        ParamGenerator<ParamType> generator((*gen_it->generator)());
        ParamNameGeneratorFunc* name_func = gen_it->name_func;
        const char* file = gen_it->file;
        int line = gen_it->line;

        std::string test_case_name;
        if ( !instantiation_name.empty() )
          test_case_name = instantiation_name + "/";
        test_case_name += test_info->test_case_base_name;

        size_t i = 0;
        std::set<std::string> test_param_names;
        for (typename ParamGenerator<ParamType>::iterator param_it =
                 generator.begin();
             param_it != generator.end(); ++param_it, ++i) {
          Message test_name_stream;

          std::string param_name = name_func(
              TestParamInfo<ParamType>(*param_it, i));

          GTEST_CHECK_(IsValidParamName(param_name))
              « "Parameterized test name '" « param_name
              « "' is invalid, in " « file
              « " line " « line « std::endl;

          GTEST_CHECK_(test_param_names.count(param_name) == 0)
              « "Duplicate parameterized test name '" « param_name
              « "', in " « file « " line " « line « std::endl;

          test_param_names.insert(param_name);

          test_name_stream « test_info->test_base_name « "/" « param_name;
          MakeAndRegisterTestInfo(
              test_case_name.c_str(),
              test_name_stream.GetString().c_str(),
              NULL,  // No type parameter.
```

```
              PrintToString(*param_it).c_str(),
              code_location_,
              GetTestCaseTypeId(),
              TestCase::SetUpTestCase,
              TestCase::TearDownTestCase,
              test_info->test_meta_factory->CreateTestFactory(*param_it));
        }  // for param_it
      }  // for gen_it
    }  // for test_it
  }  // RegisterTests

 private:
  // LocalTestInfo structure keeps information about a single test registered
  // with TEST_P macro.
  struct TestInfo {
    TestInfo(const char* a_test_case_base_name,
             const char* a_test_base_name,
             TestMetaFactoryBase<ParamType>* a_test_meta_factory) :
        test_case_base_name(a_test_case_base_name),
        test_base_name(a_test_base_name),
        test_meta_factory(a_test_meta_factory) {}

    const std::string test_case_base_name;
    const std::string test_base_name;
    const scoped_ptr<TestMetaFactoryBase<ParamType> > test_meta_factory;
  };
  typedef ::std::vector<linked_ptr<TestInfo> > TestInfoContainer;
  // Records data received from INSTANTIATE_TEST_CASE_P macros:
  //  <Instantiation name, Sequence generator creation function,
  //     Name generator function, Source file, Source line>
  struct InstantiationInfo {
      InstantiationInfo(const std::string &name_in,
                        GeneratorCreationFunc* generator_in,
                        ParamNameGeneratorFunc* name_func_in,
                        const char* file_in,
                        int line_in)
          : name(name_in),
            generator(generator_in),
            name_func(name_func_in),
            file(file_in),
            line(line_in) {}

      std::string name;
      GeneratorCreationFunc* generator;
      ParamNameGeneratorFunc* name_func;
      const char* file;
      int line;
  };
  typedef ::std::vector<InstantiationInfo> InstantiationContainer;

  static bool IsValidParamName(const std::string& name) {
    // Check for empty string
    if (name.empty())
      return false;

    // Check for invalid characters
    for (std::string::size_type index = 0; index < name.size(); ++index) {
      if (!isalnum(name[index]) && name[index] != '_')
        return false;
    }

    return true;
  }

  const std::string test_case_name_;
  CodeLocation code_location_;
  TestInfoContainer tests_;
  InstantiationContainer instantiations_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(ParameterizedTestCaseInfo);
};  // class ParameterizedTestCaseInfo

// INTERNAL IMPLEMENTATION - DO NOT USE IN USER CODE.
//
// ParameterizedTestCaseRegistry contains a map of ParameterizedTestCaseInfoBase
// classes accessed by test case names. TEST_P and INSTANTIATE_TEST_CASE_P
// macros use it to locate their corresponding ParameterizedTestCaseInfo
// descriptors.
class ParameterizedTestCaseRegistry {
 public:
  ParameterizedTestCaseRegistry() {}
  ~ParameterizedTestCaseRegistry() {
    for (TestCaseInfoContainer::iterator it = test_case_infos_.begin();
         it != test_case_infos_.end(); ++it) {
      delete *it;
    }
  }

  // Looks up or creates and returns a structure containing information about
  // tests and instantiations of a particular test case.
  template <class TestCase>
  ParameterizedTestCaseInfo<TestCase>* GetTestCasePatternHolder(
      const char* test_case_name,
      CodeLocation code_location) {
    ParameterizedTestCaseInfo<TestCase>* typed_test_info = NULL;
    for (TestCaseInfoContainer::iterator it = test_case_infos_.begin();
         it != test_case_infos_.end(); ++it) {
      if ((*it)->GetTestCaseName() == test_case_name) {
        if ((*it)->GetTestCaseTypeId() != GetTypeId<TestCase>()) {
          // Complain about incorrect usage of Google Test facilities
          // and terminate the program since we cannot guaranty correct
          // test case setup and tear-down in this case.
          ReportInvalidTestCaseType(test_case_name, code_location);
          posix::Abort();
        } else {
          // At this point we are sure that the object we found is of the same
          // type we are looking for, so we downcast it to that type
          // without further checks.
          typed_test_info = CheckedDowncastToActualType<
              ParameterizedTestCaseInfo<TestCase> >(*it);
        }
        break;
      }
    }
    if (typed_test_info == NULL) {
      typed_test_info = new ParameterizedTestCaseInfo<TestCase>(
          test_case_name, code_location);
      test_case_infos_.push_back(typed_test_info);
    }
    return typed_test_info;
  }
  void RegisterTests() {
    for (TestCaseInfoContainer::iterator it = test_case_infos_.begin();
         it != test_case_infos_.end(); ++it) {
      (*it)->RegisterTests();
    }
  }

 private:
  typedef ::std::vector<ParameterizedTestCaseInfoBase*> TestCaseInfoContainer;

  TestCaseInfoContainer test_case_infos_;

  GTEST_DISALLOW_COPY_AND_ASSIGN_(ParameterizedTestCaseRegistry);
};

}  // namespace internal
}  // namespace testing

#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.

```
// Copyright 2015, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// The Google C++ Testing and Mocking Framework (Google Test)
//
// This header file defines the GTEST_OS_* macro.
// It is separate from gtest-port.h so that custom/gtest-port.h can include it.

#ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PORT_ARCH_H_
#define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PORT_ARCH_H_

// Determines the platform on which Google Test is compiled.
#ifdef __CYGWIN__
# define GTEST_OS_CYGWIN 1
#elif defined __SYMBIAN32__
# define GTEST_OS_SYMBIAN 1
#elif defined _WIN32
# define GTEST_OS_WINDOWS 1
# ifdef _WIN32_WCE
#  define GTEST_OS_WINDOWS_MOBILE 1
# elif defined(__MINGW__) || defined(__MINGW32__)
#  define GTEST_OS_WINDOWS_MINGW 1
# elif defined(WINAPI_FAMILY)
#  include <winapifamily.h>
#  if WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_DESKTOP)
#   define GTEST_OS_WINDOWS_DESKTOP 1
#  elif WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_PHONE_APP)
#   define GTEST_OS_WINDOWS_PHONE 1
#  elif WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_APP)
#   define GTEST_OS_WINDOWS_RT 1
#  elif WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_TV_TITLE)
#   define GTEST_OS_WINDOWS_PHONE 1
#   define GTEST_OS_WINDOWS_TV_TITLE 1
#  else
    // WINAPI_FAMILY defined but no known partition matched.
    // Default to desktop.
#   define GTEST_OS_WINDOWS_DESKTOP 1
#  endif
# else
#  define GTEST_OS_WINDOWS_DESKTOP 1
# endif  // _WIN32_WCE
#elif defined __APPLE__
# define GTEST_OS_MAC 1
# if TARGET_OS_IPHONE
#  define GTEST_OS_IOS 1
# endif
#elif defined __FreeBSD__
# define GTEST_OS_FREEBSD 1
#elif defined __Fuchsia__
# define GTEST_OS_FUCHSIA 1
#elif defined __linux__
# define GTEST_OS_LINUX 1
# if defined __ANDROID__
#  define GTEST_OS_LINUX_ANDROID 1
# endif
#elif defined __MVS__
# define GTEST_OS_ZOS 1
#elif defined(__sun) && defined(__SVR4)
# define GTEST_OS_SOLARIS 1
#elif defined(_AIX)
# define GTEST_OS_AIX 1
#elif defined(__hpux)
# define GTEST_OS_HPUX 1
#elif defined __native_client__
# define GTEST_OS_NACL 1
#elif defined __NetBSD__
# define GTEST_OS_NETBSD 1
#elif defined __OpenBSD__
# define GTEST_OS_OPENBSD 1
#elif defined __QNX__
# define GTEST_OS_QNX 1
#endif  // __CYGWIN__

#endif  // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_PORT_ARCH_H_
```

9.47 Dokumentacja pliku packages/Microsoft.googletest.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.

```
// Copyright 2005, Google Inc.
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// The Google C++ Testing and Mocking Framework (Google Test)
//
// This header file declares the String class and functions used internally by
// Google Test.  They are subject to change without notice. They should not used
// by code external to Google Test.
//
// This header file is #included by gtest-internal.h.
// It should not be #included by other files.

// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_STRING_H_
#define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_STRING_H_

#ifdef __BORLANDC__
// string.h is not guaranteed to provide strcpy on C++ Builder.
# include <mem.h>
#endif

#include <string.h>
#include <string>

#include "gtest/internal/gtest-port.h"

namespace testing {
namespace internal {

// String - an abstract class holding static string utilities.
class GTEST_API_ String {
 public:
  // Static utility methods

  // Clones a 0-terminated C string, allocating memory using new.  The
  // caller is responsible for deleting the return value using
  // delete[].  Returns the cloned string, or NULL if the input is
  // NULL.
  //
  // This is different from strdup() in string.h, which allocates
  // memory using malloc().
  static const char* CloneCString(const char* c_str);

#if GTEST_OS_WINDOWS_MOBILE
  // Windows CE does not have the 'ANSI' versions of Win32 APIs. To be
  // able to pass strings to Win32 APIs on CE we need to convert them
  // to 'Unicode', UTF-16.

  // Creates a UTF-16 wide string from the given ANSI string, allocating
  // memory using new. The caller is responsible for deleting the return
  // value using delete[]. Returns the wide string, or NULL if the
  // input is NULL.
  //
  // The wide string is created using the ANSI codepage (CP_ACP) to
  // match the behaviour of the ANSI versions of Win32 calls and the
  // C runtime.
  static LPCWSTR AnsiToUtf16(const char* c_str);

  // Creates an ANSI string from the given wide string, allocating
  // memory using new. The caller is responsible for deleting the return
  // value using delete[]. Returns the ANSI string, or NULL if the
  // input is NULL.
  //
  // The returned string is created using the ANSI codepage (CP_ACP) to
  // match the behaviour of the ANSI versions of Win32 calls and the
  // C runtime.
  static const char* Utf16ToAnsi(LPCWSTR utf16_str);
#endif

  // Compares two C strings.  Returns true iff they have the same content.
  //
  // Unlike strcmp(), this function can handle NULL argument(s).  A
  // NULL C string is considered different to any non-NULL C string,
  // including the empty string.
  static bool CStringEquals(const char* lhs, const char* rhs);

  // Converts a wide C string to a String using the UTF-8 encoding.
  // NULL will be converted to "(null)".  If an error occurred during
  // the conversion, "(failed to convert from wide string)" is
  // returned.
  static std::string ShowWideCString(const wchar_t* wide_c_str);

  // Compares two wide C strings.  Returns true iff they have the same
  // content.
  //
  // Unlike wcscmp(), this function can handle NULL argument(s).  A
  // NULL C string is considered different to any non-NULL C string,
  // including the empty string.
  static bool WideCStringEquals(const wchar_t* lhs, const wchar_t* rhs);

  // Compares two C strings, ignoring case.  Returns true iff they
  // have the same content.
  //
  // Unlike strcasecmp(), this function can handle NULL argument(s).
  // A NULL C string is considered different to any non-NULL C string,
  // including the empty string.
  static bool CaseInsensitiveCStringEquals(const char* lhs,
                                           const char* rhs);

  // Compares two wide C strings, ignoring case.  Returns true iff they
  // have the same content.
  //
  // Unlike wcscasecmp(), this function can handle NULL argument(s).
  // A NULL C string is considered different to any non-NULL wide C string,
  // including the empty string.
  // NB: The implementations on different platforms slightly differ.
  // On windows, this method uses _wcsicmp which compares according to LC_CTYPE
  // environment variable. On GNU platform this method uses wcscasecmp
  // which compares according to LC_CTYPE category of the current locale.
  // On MacOS X, it uses towlower, which also uses LC_CTYPE category of the
  // current locale.
  static bool CaseInsensitiveWideCStringEquals(const wchar_t* lhs,
                                               const wchar_t* rhs);

  // Returns true iff the given string ends with the given suffix, ignoring
  // case. Any string is considered to end with an empty suffix.
  static bool EndsWithCaseInsensitive(
      const std::string& str, const std::string& suffix);

  // Formats an int value as "%02d".
  static std::string FormatIntWidth2(int value);  // "%02d" for width == 2

  // Formats an int value as "%X".
  static std::string FormatHexInt(int value);

  // Formats a byte as "%02X".
  static std::string FormatByte(unsigned char value);

 private:
  String();  // Not meant to be instantiated.
};  // class String

// Gets the content of the stringstream's buffer as an std::string.  Each '\0'
// character in the buffer is replaced with "\\0".
GTEST_API_ std::string StringStreamToString(::std::stringstream* stream);

}  // namespace internal
}  // namespace testing

#endif  // GTEST_INCLUDE_GTEST_INTERNAL_GTEST_STRING_H_
```

9.49 Dokumentacja pliku packages/Microsoft.googletest.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, 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, 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, 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.

```
// This file was GENERATED by command:
//     pump.py gtest-tuple.h.pump
// DO NOT EDIT BY HAND!!!

// Copyright 2009 Google Inc.
// All Rights Reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
```

```
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


// Implements a subset of TR1 tuple needed by Google Test and Google Mock.

// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_TUPLE_H_
#define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_TUPLE_H_

#include <utility>  // For ::std::pair.

// The compiler used in Symbian has a bug that prevents us from declaring the
// tuple template as a friend (it complains that tuple is redefined).  This
// bypasses the bug by declaring the members that should otherwise be
// private as public.
// Sun Studio versions < 12 also have the above bug.
#if defined(__SYMBIAN32__) || (defined(__SUNPRO_CC) && __SUNPRO_CC < 0x590)
# define GTEST_DECLARE_TUPLE_AS_FRIEND_ public:
#else
# define GTEST_DECLARE_TUPLE_AS_FRIEND_ \
    template <GTEST_10_TYPENAMES_(U)> friend class tuple; \
   private:
#endif

// Visual Studio 2010, 2012, and 2013 define symbols in std::tr1 that conflict
// with our own definitions. Therefore using our own tuple does not work on
// those compilers.
#if defined(_MSC_VER) && _MSC_VER >= 1600  /* 1600 is Visual Studio 2010 */
# error "gtest's tuple doesn't compile on Visual Studio 2010 or later. \
GTEST_USE_OWN_TR1_TUPLE must be set to 0 on those compilers."
#endif

// GTEST_n_TUPLE_(T) is the type of an n-tuple.
#define GTEST_0_TUPLE_(T) tuple<>
#define GTEST_1_TUPLE_(T) tuple<T##0, void, void, void, void, void, void, \
    void, void, void>
#define GTEST_2_TUPLE_(T) tuple<T##0, T##1, void, void, void, void, void, \
    void, void, void>
#define GTEST_3_TUPLE_(T) tuple<T##0, T##1, T##2, void, void, void, void, \
    void, void, void>
#define GTEST_4_TUPLE_(T) tuple<T##0, T##1, T##2, T##3, void, void, void, \
    void, void, void>
#define GTEST_5_TUPLE_(T) tuple<T##0, T##1, T##2, T##3, T##4, void, void, \
    void, void, void>
#define GTEST_6_TUPLE_(T) tuple<T##0, T##1, T##2, T##3, T##4, T##5, void, \
    void, void, void>
#define GTEST_7_TUPLE_(T) tuple<T##0, T##1, T##2, T##3, T##4, T##5, T##6, \
    void, void, void>
#define GTEST_8_TUPLE_(T) tuple<T##0, T##1, T##2, T##3, T##4, T##5, T##6, \
    T##7, void, void>
#define GTEST_9_TUPLE_(T) tuple<T##0, T##1, T##2, T##3, T##4, T##5, T##6, \
    T##7, T##8, void>
#define GTEST_10_TUPLE_(T) tuple<T##0, T##1, T##2, T##3, T##4, T##5, T##6, \
    T##7, T##8, T##9>

// GTEST_n_TYPENAMES_(T) declares a list of n typenames.
#define GTEST_0_TYPENAMES_(T)
#define GTEST_1_TYPENAMES_(T) typename T##0
#define GTEST_2_TYPENAMES_(T) typename T##0, typename T##1
#define GTEST_3_TYPENAMES_(T) typename T##0, typename T##1, typename T##2
#define GTEST_4_TYPENAMES_(T) typename T##0, typename T##1, typename T##2, \
    typename T##3
#define GTEST_5_TYPENAMES_(T) typename T##0, typename T##1, typename T##2, \
    typename T##3, typename T##4
#define GTEST_6_TYPENAMES_(T) typename T##0, typename T##1, typename T##2, \
    typename T##3, typename T##4, typename T##5
#define GTEST_7_TYPENAMES_(T) typename T##0, typename T##1, typename T##2, \
    typename T##3, typename T##4, typename T##5, typename T##6
#define GTEST_8_TYPENAMES_(T) typename T##0, typename T##1, typename T##2, \
    typename T##3, typename T##4, typename T##5, typename T##6, typename T##7
#define GTEST_9_TYPENAMES_(T) 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
#define GTEST_10_TYPENAMES_(T) 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
```

```

// In theory, defining stuff in the ::std namespace is undefined
// behavior.  We can do this as we are playing the role of a standard
// library vendor.
namespace std {
namespace tr1 {

template <typename T0 = void, typename T1 = void, typename T2 = void,
    typename T3 = void, typename T4 = void, typename T5 = void,
    typename T6 = void, typename T7 = void, typename T8 = void,
    typename T9 = void>
class tuple;

// Anything in namespace gtest_internal is Google Test's INTERNAL
// IMPLEMENTATION DETAIL and MUST NOT BE USED DIRECTLY in user code.
namespace gtest_internal {

// ByRef<T>::type is T if T is a reference; otherwise it's const T&.
template <typename T>
struct ByRef { typedef const T& type; };  // NOLINT
template <typename T>
struct ByRef<T&> { typedef T& type; };  // NOLINT

// A handy wrapper for ByRef.
#define GTEST_BY_REF_(T) typename ::std::tr1::gtest_internal::ByRef<T>::type

// AddRef<T>::type is T if T is a reference; otherwise it's T&.  This
// is the same as tr1::add_reference<T>::type.
template <typename T>
struct AddRef { typedef T& type; };  // NOLINT
template <typename T>
struct AddRef<T&> { typedef T& type; };  // NOLINT

// A handy wrapper for AddRef.
#define GTEST_ADD_REF_(T) typename ::std::tr1::gtest_internal::AddRef<T>::type

// A helper for implementing get<k>().
template <int k> class Get;

// A helper for implementing tuple_element<k, T>.  kIndexValid is true
// iff k < the number of fields in tuple type T.
template <bool kIndexValid, int kIndex, class Tuple>
struct TupleElement;

template <GTEST_10_TYPENAMES_(T)>
struct TupleElement<true, 0, GTEST_10_TUPLE_(T) > {
  typedef T0 type;
};

template <GTEST_10_TYPENAMES_(T)>
struct TupleElement<true, 1, GTEST_10_TUPLE_(T) > {
  typedef T1 type;
};

template <GTEST_10_TYPENAMES_(T)>
struct TupleElement<true, 2, GTEST_10_TUPLE_(T) > {
  typedef T2 type;
};

template <GTEST_10_TYPENAMES_(T)>
struct TupleElement<true, 3, GTEST_10_TUPLE_(T) > {
  typedef T3 type;
};

template <GTEST_10_TYPENAMES_(T)>
struct TupleElement<true, 4, GTEST_10_TUPLE_(T) > {
  typedef T4 type;
};

template <GTEST_10_TYPENAMES_(T)>
struct TupleElement<true, 5, GTEST_10_TUPLE_(T) > {
  typedef T5 type;
};

template <GTEST_10_TYPENAMES_(T)>
struct TupleElement<true, 6, GTEST_10_TUPLE_(T) > {
  typedef T6 type;
};

template <GTEST_10_TYPENAMES_(T)>
struct TupleElement<true, 7, GTEST_10_TUPLE_(T) > {
  typedef T7 type;
};

template <GTEST_10_TYPENAMES_(T)>
struct TupleElement<true, 8, GTEST_10_TUPLE_(T) > {
  typedef T8 type;
};

template <GTEST_10_TYPENAMES_(T)>
struct TupleElement<true, 9, GTEST_10_TUPLE_(T) > {
  typedef T9 type;
};

}  // namespace gtest_internal

template <>
class tuple<> {
 public:
  tuple() {}
  tuple(const tuple& /* t */)  {}
  tuple& operator=(const tuple& /* t */) { return *this; }
};

template <GTEST_1_TYPENAMES_(T)>
class GTEST_1_TUPLE_(T) {
 public:
  template <int k> friend class gtest_internal::Get;

  tuple() : f0_() {}

  explicit tuple(GTEST_BY_REF_(T0) f0) : f0_(f0) {}

  tuple(const tuple& t) : f0_(t.f0_) {}

  template <GTEST_1_TYPENAMES_(U)>
  tuple(const GTEST_1_TUPLE_(U)& t) : f0_(t.f0_) {}

  tuple& operator=(const tuple& t) { return CopyFrom(t); }

  template <GTEST_1_TYPENAMES_(U)>
  tuple& operator=(const GTEST_1_TUPLE_(U)& t) {
    return CopyFrom(t);
  }

  GTEST_DECLARE_TUPLE_AS_FRIEND_

  template <GTEST_1_TYPENAMES_(U)>
  tuple& CopyFrom(const GTEST_1_TUPLE_(U)& t) {
    f0_ = t.f0_;
    return *this;
  }

  T0 f0_;
};

template <GTEST_2_TYPENAMES_(T)>
class GTEST_2_TUPLE_(T) {
 public:
  template <int k> friend class gtest_internal::Get;

  tuple() : f0_(), f1_() {}

  explicit tuple(GTEST_BY_REF_(T0) f0, GTEST_BY_REF_(T1) f1) : f0_(f0),
      f1_(f1) {}

  tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_) {}

  template <GTEST_2_TYPENAMES_(U)>
  tuple(const GTEST_2_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_) {}
  template <typename U0, typename U1>
  tuple(const ::std::pair<U0, U1>& p) : f0_(p.first), f1_(p.second) {}

  tuple& operator=(const tuple& t) { return CopyFrom(t); }

  template <GTEST_2_TYPENAMES_(U)>
  tuple& operator=(const GTEST_2_TUPLE_(U)& t) {
    return CopyFrom(t);
  }
  template <typename U0, typename U1>
  tuple& operator=(const ::std::pair<U0, U1>& p) {
    f0_ = p.first;
    f1_ = p.second;
    return *this;
  }

  GTEST_DECLARE_TUPLE_AS_FRIEND_

  template <GTEST_2_TYPENAMES_(U)>
  tuple& CopyFrom(const GTEST_2_TUPLE_(U)& t) {
    f0_ = t.f0_;
    f1_ = t.f1_;
    return *this;
  }

  T0 f0_;
  T1 f1_;
};

template <GTEST_3_TYPENAMES_(T)>
class GTEST_3_TUPLE_(T) {
 public:
  template <int k> friend class gtest_internal::Get;

  tuple() : f0_(), f1_(), f2_() {}

  explicit tuple(GTEST_BY_REF_(T0) f0, GTEST_BY_REF_(T1) f1,
      GTEST_BY_REF_(T2) f2) : f0_(f0), f1_(f1), f2_(f2) {}

  tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_) {}

  template <GTEST_3_TYPENAMES_(U)>
  tuple(const GTEST_3_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_) {}

  tuple& operator=(const tuple& t) { return CopyFrom(t); }

  template <GTEST_3_TYPENAMES_(U)>
  tuple& operator=(const GTEST_3_TUPLE_(U)& t) {
    return CopyFrom(t);
  }

  GTEST_DECLARE_TUPLE_AS_FRIEND_

  template <GTEST_3_TYPENAMES_(U)>
  tuple& CopyFrom(const GTEST_3_TUPLE_(U)& t) {
    f0_ = t.f0_;
    f1_ = t.f1_;
    f2_ = t.f2_;
    return *this;
  }

  T0 f0_;
  T1 f1_;
  T2 f2_;
};

template <GTEST_4_TYPENAMES_(T)>
class GTEST_4_TUPLE_(T) {
 public:
  template <int k> friend class gtest_internal::Get;

  tuple() : f0_(), f1_(), f2_(), f3_() {}

  explicit tuple(GTEST_BY_REF_(T0) f0, GTEST_BY_REF_(T1) f1,
      GTEST_BY_REF_(T2) f2, GTEST_BY_REF_(T3) f3) : f0_(f0), f1_(f1), f2_(f2),
      f3_(f3) {}

  tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_), f3_(t.f3_) {}

  template <GTEST_4_TYPENAMES_(U)>
  tuple(const GTEST_4_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_),
      f3_(t.f3_) {}

  tuple& operator=(const tuple& t) { return CopyFrom(t); }

  template <GTEST_4_TYPENAMES_(U)>
  tuple& operator=(const GTEST_4_TUPLE_(U)& t) {
    return CopyFrom(t);
  }

  GTEST_DECLARE_TUPLE_AS_FRIEND_

  template <GTEST_4_TYPENAMES_(U)>
  tuple& CopyFrom(const GTEST_4_TUPLE_(U)& t) {
    f0_ = t.f0_;
    f1_ = t.f1_;
    f2_ = t.f2_;
    f3_ = t.f3_;
    return *this;
  }

  T0 f0_;
  T1 f1_;
  T2 f2_;
  T3 f3_;
};

template <GTEST_5_TYPENAMES_(T)>
class GTEST_5_TUPLE_(T) {
 public:
  template <int k> friend class gtest_internal::Get;

  tuple() : f0_(), f1_(), f2_(), f3_(), f4_() {}

  explicit 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) : f0_(f0), f1_(f1), f2_(f2), f3_(f3), f4_(f4) {}

  tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_), f3_(t.f3_),
      f4_(t.f4_) {}

  template <GTEST_5_TYPENAMES_(U)>
  tuple(const GTEST_5_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_),
      f3_(t.f3_), f4_(t.f4_) {}

  tuple& operator=(const tuple& t) { return CopyFrom(t); }

  template <GTEST_5_TYPENAMES_(U)>
  tuple& operator=(const GTEST_5_TUPLE_(U)& t) {
    return CopyFrom(t);
  }

  GTEST_DECLARE_TUPLE_AS_FRIEND_

  template <GTEST_5_TYPENAMES_(U)>
  tuple& CopyFrom(const GTEST_5_TUPLE_(U)& t) {
    f0_ = t.f0_;
    f1_ = t.f1_;
    f2_ = t.f2_;
    f3_ = t.f3_;
    f4_ = t.f4_;
    return *this;
  }

  T0 f0_;
  T1 f1_;
  T2 f2_;
  T3 f3_;
  T4 f4_;
};

template <GTEST_6_TYPENAMES_(T)>
class GTEST_6_TUPLE_(T) {
 public:
  template <int k> friend class gtest_internal::Get;

  tuple() : f0_(), f1_(), f2_(), f3_(), f4_(), f5_() {}

  explicit 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) : f0_(f0), f1_(f1), f2_(f2), f3_(f3), f4_(f4),
      f5_(f5) {}

  tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_), f3_(t.f3_),
      f4_(t.f4_), f5_(t.f5_) {}

  template <GTEST_6_TYPENAMES_(U)>
  tuple(const GTEST_6_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_),
      f3_(t.f3_), f4_(t.f4_), f5_(t.f5_) {}

  tuple& operator=(const tuple& t) { return CopyFrom(t); }

  template <GTEST_6_TYPENAMES_(U)>
  tuple& operator=(const GTEST_6_TUPLE_(U)& t) {
    return CopyFrom(t);
  }

  GTEST_DECLARE_TUPLE_AS_FRIEND_

  template <GTEST_6_TYPENAMES_(U)>
  tuple& CopyFrom(const GTEST_6_TUPLE_(U)& t) {
    f0_ = t.f0_;
    f1_ = t.f1_;
    f2_ = t.f2_;
    f3_ = t.f3_;
    f4_ = t.f4_;
    f5_ = t.f5_;
    return *this;
  }

  T0 f0_;
  T1 f1_;
  T2 f2_;
  T3 f3_;
  T4 f4_;
  T5 f5_;
};

```

```
template <GTEST_7_TYPENAMES_(T)>
class GTEST_7_TUPLE_(T) {
 public:
  template <int k> friend class gtest_internal::Get;

  tuple() : f0_(), f1_(), f2_(), f3_(), f4_(), f5_(), f6_() {}

  explicit 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) : f0_(f0), f1_(f1), f2_(f2),
      f3_(f3), f4_(f4), f5_(f5), f6_(f6) {}

  tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_), f3_(t.f3_),
      f4_(t.f4_), f5_(t.f5_), f6_(t.f6_) {}

  template <GTEST_7_TYPENAMES_(U)>
  tuple(const GTEST_7_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_),
      f3_(t.f3_), f4_(t.f4_), f5_(t.f5_), f6_(t.f6_) {}

  tuple& operator=(const tuple& t) { return CopyFrom(t); }

  template <GTEST_7_TYPENAMES_(U)>
  tuple& operator=(const GTEST_7_TUPLE_(U)& t) {
    return CopyFrom(t);
  }

  GTEST_DECLARE_TUPLE_AS_FRIEND_

  template <GTEST_7_TYPENAMES_(U)>
  tuple& CopyFrom(const GTEST_7_TUPLE_(U)& t) {
    f0_ = t.f0_;
    f1_ = t.f1_;
    f2_ = t.f2_;
    f3_ = t.f3_;
    f4_ = t.f4_;
    f5_ = t.f5_;
    f6_ = t.f6_;
    return *this;
  }

  T0 f0_;
  T1 f1_;
  T2 f2_;
  T3 f3_;
  T4 f4_;
  T5 f5_;
  T6 f6_;
};

template <GTEST_8_TYPENAMES_(T)>
class GTEST_8_TUPLE_(T) {
 public:
  template <int k> friend class gtest_internal::Get;

  tuple() : f0_(), f1_(), f2_(), f3_(), f4_(), f5_(), f6_(), f7_() {}

  explicit 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) : f0_(f0), f1_(f1), f2_(f2), f3_(f3), f4_(f4),
      f5_(f5), f6_(f6), f7_(f7) {}

  tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_), f3_(t.f3_),
      f4_(t.f4_), f5_(t.f5_), f6_(t.f6_), f7_(t.f7_) {}

  template <GTEST_8_TYPENAMES_(U)>
  tuple(const GTEST_8_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_),
      f3_(t.f3_), f4_(t.f4_), f5_(t.f5_), f6_(t.f6_), f7_(t.f7_) {}

  tuple& operator=(const tuple& t) { return CopyFrom(t); }

  template <GTEST_8_TYPENAMES_(U)>
  tuple& operator=(const GTEST_8_TUPLE_(U)& t) {
    return CopyFrom(t);
  }

  GTEST_DECLARE_TUPLE_AS_FRIEND_

  template <GTEST_8_TYPENAMES_(U)>
  tuple& CopyFrom(const GTEST_8_TUPLE_(U)& t) {
    f0_ = t.f0_;
    f1_ = t.f1_;
    f2_ = t.f2_;
    f3_ = t.f3_;
    f4_ = t.f4_;
    f5_ = t.f5_;
    f6_ = t.f6_;
    f7_ = t.f7_;
    return *this;
  }

  T0 f0_;
  T1 f1_;
  T2 f2_;
  T3 f3_;
  T4 f4_;
  T5 f5_;
  T6 f6_;
  T7 f7_;
};

template <GTEST_9_TYPENAMES_(T)>
class GTEST_9_TUPLE_(T) {
 public:
  template <int k> friend class gtest_internal::Get;

  tuple() : f0_(), f1_(), f2_(), f3_(), f4_(), f5_(), f6_(), f7_(), f8_() {}

  explicit 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) : f0_(f0), f1_(f1), f2_(f2), f3_(f3), f4_(f4),
      f5_(f5), f6_(f6), f7_(f7), f8_(f8) {}

  tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_), f3_(t.f3_),
      f4_(t.f4_), f5_(t.f5_), f6_(t.f6_), f7_(t.f7_), f8_(t.f8_) {}

  template <GTEST_9_TYPENAMES_(U)>
  tuple(const GTEST_9_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_),
      f3_(t.f3_), f4_(t.f4_), f5_(t.f5_), f6_(t.f6_), f7_(t.f7_), f8_(t.f8_) {}

  tuple& operator=(const tuple& t) { return CopyFrom(t); }

  template <GTEST_9_TYPENAMES_(U)>
  tuple& operator=(const GTEST_9_TUPLE_(U)& t) {
    return CopyFrom(t);
  }

  GTEST_DECLARE_TUPLE_AS_FRIEND_

  template <GTEST_9_TYPENAMES_(U)>
  tuple& CopyFrom(const GTEST_9_TUPLE_(U)& t) {
    f0_ = t.f0_;
    f1_ = t.f1_;
    f2_ = t.f2_;
    f3_ = t.f3_;
    f4_ = t.f4_;
    f5_ = t.f5_;
    f6_ = t.f6_;
    f7_ = t.f7_;
    f8_ = t.f8_;
    return *this;
  }

  T0 f0_;
  T1 f1_;
  T2 f2_;
  T3 f3_;
  T4 f4_;
  T5 f5_;
  T6 f6_;
  T7 f7_;
  T8 f8_;
};

template <GTEST_10_TYPENAMES_(T)>
class tuple {
 public:
  template <int k> friend class gtest_internal::Get;

  tuple() : f0_(), f1_(), f2_(), f3_(), f4_(), f5_(), f6_(), f7_(), f8_(),
      f9_() {}

  explicit 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) : f0_(f0), f1_(f1), f2_(f2),
      f3_(f3), f4_(f4), f5_(f5), f6_(f6), f7_(f7), f8_(f8), f9_(f9) {}

  tuple(const tuple& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_), f3_(t.f3_),
      f4_(t.f4_), f5_(t.f5_), f6_(t.f6_), f7_(t.f7_), f8_(t.f8_), f9_(t.f9_) {}

  template <GTEST_10_TYPENAMES_(U)>
  tuple(const GTEST_10_TUPLE_(U)& t) : f0_(t.f0_), f1_(t.f1_), f2_(t.f2_),
      f3_(t.f3_), f4_(t.f4_), f5_(t.f5_), f6_(t.f6_), f7_(t.f7_), f8_(t.f8_),
      f9_(t.f9_) {}

  tuple& operator=(const tuple& t) { return CopyFrom(t); }

  template <GTEST_10_TYPENAMES_(U)>
  tuple& operator=(const GTEST_10_TUPLE_(U)& t) {
    return CopyFrom(t);
  }

  GTEST_DECLARE_TUPLE_AS_FRIEND_

  template <GTEST_10_TYPENAMES_(U)>
  tuple& CopyFrom(const GTEST_10_TUPLE_(U)& t) {
    f0_ = t.f0_;
    f1_ = t.f1_;
    f2_ = t.f2_;
    f3_ = t.f3_;
    f4_ = t.f4_;
    f5_ = t.f5_;
    f6_ = t.f6_;
    f7_ = t.f7_;
    f8_ = t.f8_;
    f9_ = t.f9_;
    return *this;
  }

  T0 f0_;
  T1 f1_;
  T2 f2_;
  T3 f3_;
  T4 f4_;
  T5 f5_;
  T6 f6_;
  T7 f7_;
  T8 f8_;
  T9 f9_;
};

// 6.1.3.2 Tuple creation functions.

// Known limitations: we don't support passing an
// std::tr1::reference_wrapper<T> to make_tuple().  And we don't
// implement tie().

inline tuple<> make_tuple() { return tuple<>(); }

template <GTEST_1_TYPENAMES_(T)>
inline GTEST_1_TUPLE_(T) make_tuple(const T0& f0) {
  return GTEST_1_TUPLE_(T)(f0);
}

template <GTEST_2_TYPENAMES_(T)>
inline GTEST_2_TUPLE_(T) make_tuple(const T0& f0, const T1& f1) {
  return GTEST_2_TUPLE_(T)(f0, f1);
}

template <GTEST_3_TYPENAMES_(T)>
inline GTEST_3_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2) {
  return GTEST_3_TUPLE_(T)(f0, f1, f2);
}

template <GTEST_4_TYPENAMES_(T)>
inline GTEST_4_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2,
    const T3& f3) {
  return GTEST_4_TUPLE_(T)(f0, f1, f2, f3);
}

template <GTEST_5_TYPENAMES_(T)>
inline GTEST_5_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2,
    const T3& f3, const T4& f4) {
  return GTEST_5_TUPLE_(T)(f0, f1, f2, f3, f4);
}

template <GTEST_6_TYPENAMES_(T)>
inline GTEST_6_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2,
    const T3& f3, const T4& f4, const T5& f5) {
  return GTEST_6_TUPLE_(T)(f0, f1, f2, f3, f4, f5);
}

template <GTEST_7_TYPENAMES_(T)>
inline GTEST_7_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2,
    const T3& f3, const T4& f4, const T5& f5, const T6& f6) {
  return GTEST_7_TUPLE_(T)(f0, f1, f2, f3, f4, f5, f6);
}

template <GTEST_8_TYPENAMES_(T)>
```

```
inline GTEST_8_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2,
    const T3& f3, const T4& f4, const T5& f5, const T6& f6, const T7& f7) {
  return GTEST_8_TUPLE_(T)(f0, f1, f2, f3, f4, f5, f6, f7);
}

template <GTEST_9_TYPENAMES_(T)>
inline GTEST_9_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2,
    const T3& f3, const T4& f4, const T5& f5, const T6& f6, const T7& f7,
    const T8& f8) {
  return GTEST_9_TUPLE_(T)(f0, f1, f2, f3, f4, f5, f6, f7, f8);
}

template <GTEST_10_TYPENAMES_(T)>
inline GTEST_10_TUPLE_(T) make_tuple(const T0& f0, const T1& f1, const T2& f2,
    const T3& f3, const T4& f4, const T5& f5, const T6& f6, const T7& f7,
    const T8& f8, const T9& f9) {
  return GTEST_10_TUPLE_(T)(f0, f1, f2, f3, f4, f5, f6, f7, f8, f9);
}

// 6.1.3.3 Tuple helper classes.

template <typename Tuple> struct tuple_size;

template <GTEST_0_TYPENAMES_(T)>
struct tuple_size<GTEST_0_TUPLE_(T) > {
  static const int value = 0;
};

template <GTEST_1_TYPENAMES_(T)>
struct tuple_size<GTEST_1_TUPLE_(T) > {
  static const int value = 1;
};

template <GTEST_2_TYPENAMES_(T)>
struct tuple_size<GTEST_2_TUPLE_(T) > {
  static const int value = 2;
};

template <GTEST_3_TYPENAMES_(T)>
struct tuple_size<GTEST_3_TUPLE_(T) > {
  static const int value = 3;
};

template <GTEST_4_TYPENAMES_(T)>
struct tuple_size<GTEST_4_TUPLE_(T) > {
  static const int value = 4;
};

template <GTEST_5_TYPENAMES_(T)>
struct tuple_size<GTEST_5_TUPLE_(T) > {
  static const int value = 5;
};

template <GTEST_6_TYPENAMES_(T)>
struct tuple_size<GTEST_6_TUPLE_(T) > {
  static const int value = 6;
};

template <GTEST_7_TYPENAMES_(T)>
struct tuple_size<GTEST_7_TUPLE_(T) > {
  static const int value = 7;
};

template <GTEST_8_TYPENAMES_(T)>
struct tuple_size<GTEST_8_TUPLE_(T) > {
  static const int value = 8;
};

template <GTEST_9_TYPENAMES_(T)>
struct tuple_size<GTEST_9_TUPLE_(T) > {
  static const int value = 9;
};

template <GTEST_10_TYPENAMES_(T)>
struct tuple_size<GTEST_10_TUPLE_(T) > {
  static const int value = 10;
};

template <int k, class Tuple>
struct tuple_element {
  typedef typename gtest_internal::TupleElement<
      k < (tuple_size<Tuple>::value), k, Tuple>::type type;
};

#define GTEST_TUPLE_ELEMENT_(k, Tuple) typename tuple_element<k, Tuple >::type

// 6.1.3.4 Element access.
```

```

namespace gtest_internal {

template <>
class Get<0> {
 public:
  template <class Tuple>
  static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(0, Tuple))
  Field(Tuple& t) { return t.f0_; }  // NOLINT

  template <class Tuple>
  static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(0, Tuple))
  ConstField(const Tuple& t) { return t.f0_; }
};

template <>
class Get<1> {
 public:
  template <class Tuple>
  static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(1, Tuple))
  Field(Tuple& t) { return t.f1_; }  // NOLINT

  template <class Tuple>
  static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(1, Tuple))
  ConstField(const Tuple& t) { return t.f1_; }
};

template <>
class Get<2> {
 public:
  template <class Tuple>
  static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(2, Tuple))
  Field(Tuple& t) { return t.f2_; }  // NOLINT

  template <class Tuple>
  static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(2, Tuple))
  ConstField(const Tuple& t) { return t.f2_; }
};

template <>
class Get<3> {
 public:
  template <class Tuple>
  static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(3, Tuple))
  Field(Tuple& t) { return t.f3_; }  // NOLINT

  template <class Tuple>
  static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(3, Tuple))
  ConstField(const Tuple& t) { return t.f3_; }
};

template <>
class Get<4> {
 public:
  template <class Tuple>
  static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(4, Tuple))
  Field(Tuple& t) { return t.f4_; }  // NOLINT

  template <class Tuple>
  static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(4, Tuple))
  ConstField(const Tuple& t) { return t.f4_; }
};

template <>
class Get<5> {
 public:
  template <class Tuple>
  static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(5, Tuple))
  Field(Tuple& t) { return t.f5_; }  // NOLINT

  template <class Tuple>
  static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(5, Tuple))
  ConstField(const Tuple& t) { return t.f5_; }
};

template <>
class Get<6> {
 public:
  template <class Tuple>
  static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(6, Tuple))
  Field(Tuple& t) { return t.f6_; }  // NOLINT

  template <class Tuple>
  static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(6, Tuple))
  ConstField(const Tuple& t) { return t.f6_; }
};

```

```
template <>
class Get<7> {
 public:
  template <class Tuple>
  static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(7, Tuple))
  Field(Tuple& t) { return t.f7_; }  // NOLINT

  template <class Tuple>
  static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(7, Tuple))
  ConstField(const Tuple& t) { return t.f7_; }
};

template <>
class Get<8> {
 public:
  template <class Tuple>
  static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(8, Tuple))
  Field(Tuple& t) { return t.f8_; }  // NOLINT

  template <class Tuple>
  static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(8, Tuple))
  ConstField(const Tuple& t) { return t.f8_; }
};

template <>
class Get<9> {
 public:
  template <class Tuple>
  static GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(9, Tuple))
  Field(Tuple& t) { return t.f9_; }  // NOLINT

  template <class Tuple>
  static GTEST_BY_REF_(GTEST_TUPLE_ELEMENT_(9, Tuple))
  ConstField(const Tuple& t) { return t.f9_; }
};

}  // namespace gtest_internal

template <int k, GTEST_10_TYPENAMES_(T)>
GTEST_ADD_REF_(GTEST_TUPLE_ELEMENT_(k, GTEST_10_TUPLE_(T)))
get(GTEST_10_TUPLE_(T)& t) {
  return gtest_internal::Get<k>::Field(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) {
  return gtest_internal::Get<k>::ConstField(t);
}

// 6.1.3.5 Relational operators

// We only implement == and !=, as we don't have a need for the rest yet.

namespace gtest_internal {

// SameSizeTuplePrefixComparator<k, k>::Eq(t1, t2) returns true if the
// first k fields of t1 equals the first k fields of t2.
// SameSizeTuplePrefixComparator(k1, k2) would be a compiler error if
// k1 != k2.
template <int kSize1, int kSize2>
struct SameSizeTuplePrefixComparator;

template <>
struct SameSizeTuplePrefixComparator<0, 0> {
  template <class Tuple1, class Tuple2>
  static bool Eq(const Tuple1& /* t1 */, const Tuple2& /* t2 */) {
    return true;
  }
};

template <int k>
struct SameSizeTuplePrefixComparator<k, k> {
  template <class Tuple1, class Tuple2>
  static bool Eq(const Tuple1& t1, const Tuple2& t2) {
    return SameSizeTuplePrefixComparator<k - 1, k - 1>::Eq(t1, t2) &&
        ::std::tr1::get<k - 1>(t1) == ::std::tr1::get<k - 1>(t2);
  }
};

}  // namespace gtest_internal

template <GTEST_10_TYPENAMES_(T), GTEST_10_TYPENAMES_(U)>
inline bool operator==(const GTEST_10_TUPLE_(T)& t,
                       const GTEST_10_TUPLE_(U)& u) {
  return gtest_internal::SameSizeTuplePrefixComparator<
      tuple_size<GTEST_10_TUPLE_(T) >::value,
      tuple_size<GTEST_10_TUPLE_(U) >::value>::Eq(t, u);
}

template <GTEST_10_TYPENAMES_(T), GTEST_10_TYPENAMES_(U)>
inline bool operator!=(const GTEST_10_TUPLE_(T)& t,
                       const GTEST_10_TUPLE_(U)& u) { return !(t == u); }

// 6.1.4 Pairs.
// Unimplemented.

}  // namespace tr1
}  // namespace std

#undef GTEST_0_TUPLE_
#undef GTEST_1_TUPLE_
#undef GTEST_2_TUPLE_
#undef GTEST_3_TUPLE_
#undef GTEST_4_TUPLE_
#undef GTEST_5_TUPLE_
#undef GTEST_6_TUPLE_
#undef GTEST_7_TUPLE_
#undef GTEST_8_TUPLE_
#undef GTEST_9_TUPLE_
#undef GTEST_10_TUPLE_

#undef GTEST_0_TYPENAMES_
#undef GTEST_1_TYPENAMES_
#undef GTEST_2_TYPENAMES_
#undef GTEST_3_TYPENAMES_
#undef GTEST_4_TYPENAMES_
#undef GTEST_5_TYPENAMES_
#undef GTEST_6_TYPENAMES_
#undef GTEST_7_TYPENAMES_
#undef GTEST_8_TYPENAMES_
#undef GTEST_9_TYPENAMES_
#undef GTEST_10_TYPENAMES_

#undef GTEST_DECLARE_TUPLE_AS_FRIEND_
#undef GTEST_BY_REF_
#undef GTEST_ADD_REF_
#undef GTEST_TUPLE_ELEMENT_

#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.

```
// This file was GENERATED by command:
//     pump.py gtest-type-util.h.pump
// DO NOT EDIT BY HAND!!!

// Copyright 2008 Google Inc.
// All Rights Reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


// Type utilities needed for implementing typed and type-parameterized
// tests.  This file is generated by a SCRIPT.  DO NOT EDIT BY HAND!
//
// Currently we support at most 50 types in a list, and at most 50
// type-parameterized tests in one type-parameterized test case.
// Please contact googletestframework@googlegroups.com if you need
// more.

// GOOGLETEST_CM0001 DO NOT DELETE

#ifndef GTEST_INCLUDE_GTEST_INTERNAL_GTEST_TYPE_UTIL_H_
#define GTEST_INCLUDE_GTEST_INTERNAL_GTEST_TYPE_UTIL_H_

#include "gtest/internal/gtest-port.h"

// #ifdef __GNUC__ is too general here.  It is possible to use gcc without using
// libstdc++ (which is where cxxabi.h comes from).
# if GTEST_HAS_CXXABI_H_
#  include <cxxabi.h>
# elif defined(__HP_aCC)
#  include <acxx_demangle.h>
# endif  // GTEST_HASH_CXXABI_H_

namespace testing {
namespace internal {

// Canonicalizes a given name with respect to the Standard C++ Library.
// This handles removing the inline namespace within `std` that is
// used by various standard libraries (e.g., `std::__1`).  Names outside
// of namespace std are returned unmodified.
inline std::string CanonicalizeForStdLibVersioning(std::string s) {
  static const char prefix[] = "std::__";
  if (s.compare(0, strlen(prefix), prefix) == 0) {
    std::string::size_type end = s.find("::", strlen(prefix));
    if (end != s.npos) {
      // Erase everything between the initial `std` and the second `::`.
      s.erase(strlen("std"), end - strlen("std"));
    }
  }
  return s;
}

// GetTypeName<T>() returns a human-readable name of type T.
// NB: This function is also used in Google Mock, so don't move it inside of
// the typed-test-only section below.
template <typename T>
std::string GetTypeName() {
# if GTEST_HAS_RTTI
```

```

  const char* const name = typeid(T).name();
#  if GTEST_HAS_CXXABI_H_ || defined(__HP_aCC)
  int status = 0;
  // gcc's implementation of typeid(T).name() mangles the type name,
  // so we have to demangle it.
#   if GTEST_HAS_CXXABI_H_
  using abi::__cxa_demangle;
#   endif  // GTEST_HAS_CXXABI_H_
  char* const readable_name = __cxa_demangle(name, 0, 0, &status);
  const std::string name_str(status == 0 ? readable_name : name);
  free(readable_name);
  return CanonicalizeForStdLibVersioning(name_str);
#  else
  return name;
#  endif  // GTEST_HAS_CXXABI_H_ || __HP_aCC

# else

  return "<type>";

# endif  // GTEST_HAS_RTTI
}

#if GTEST_HAS_TYPED_TEST || GTEST_HAS_TYPED_TEST_P

// AssertyTypeEq<T1, T2>::type is defined iff T1 and T2 are the same
// type.  This can be used as a compile-time assertion to ensure that
// two types are equal.

template <typename T1, typename T2>
struct AssertTypeEq;

template <typename T>
struct AssertTypeEq<T, T> {
  typedef bool type;
};

// A unique type used as the default value for the arguments of class
// template Types.  This allows us to simulate variadic templates
// (e.g. Types<int>, Type<int, double>, and etc), which C++ doesn't
// support directly.
struct None {};

// The following family of struct and struct templates are used to
// represent type lists.  In particular, TypesN<T1, T2, ..., TN>
// represents a type list with N types (T1, T2, ..., and TN) in it.
// Except for Types0, every struct in the family has two member types:
// Head for the first type in the list, and Tail for the rest of the
// list.

// The empty type list.
struct Types0 {};

// Type lists of length 1, 2, 3, and so on.

template <typename T1>
struct Types1 {
  typedef T1 Head;
  typedef Types0 Tail;
};
template <typename T1, typename T2>
struct Types2 {
  typedef T1 Head;
  typedef Types1<T2> Tail;
};

template <typename T1, typename T2, typename T3>
struct Types3 {
  typedef T1 Head;
  typedef Types2<T2, T3> Tail;
};

template <typename T1, typename T2, typename T3, typename T4>
struct Types4 {
  typedef T1 Head;
  typedef Types3<T2, T3, T4> Tail;
};

template <typename T1, typename T2, typename T3, typename T4, typename T5>
struct Types5 {
  typedef T1 Head;
  typedef Types4<T2, T3, T4, T5> Tail;
};

template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6>
```

```
struct Types6 {
  typedef T1 Head;
  typedef Types5<T2, T3, T4, T5, T6> Tail;
};

template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7>
struct Types7 {
  typedef T1 Head;
  typedef Types6<T2, T3, T4, T5, T6, T7> Tail;
};

template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7, typename T8>
struct Types8 {
  typedef T1 Head;
  typedef Types7<T2, T3, T4, T5, T6, T7, T8> Tail;
};

template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7, typename T8, typename T9>
struct Types9 {
  typedef T1 Head;
  typedef Types8<T2, T3, T4, T5, T6, T7, T8, T9> Tail;
};

template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7, typename T8, typename T9, typename T10>
struct Types10 {
  typedef T1 Head;
  typedef Types9<T2, T3, T4, T5, T6, T7, T8, T9, T10> Tail;
};

template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7, typename T8, typename T9, typename T10,
    typename T11>
struct Types11 {
  typedef T1 Head;
  typedef Types10<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11> Tail;
};

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>
struct Types12 {
  typedef T1 Head;
  typedef Types11<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12> Tail;
};

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>
struct Types13 {
  typedef T1 Head;
  typedef Types12<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13> Tail;
};

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>
struct Types14 {
  typedef T1 Head;
  typedef Types13<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14> Tail;
};

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>
struct Types15 {
  typedef T1 Head;
  typedef Types14<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
      T15> Tail;
};

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>
struct Types16 {
  typedef T1 Head;
  typedef Types15<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
      T16> Tail;
};

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>
struct Types17 {
  typedef T1 Head;
  typedef Types16<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
      T16, T17> Tail;
};

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>
struct Types18 {
  typedef T1 Head;
  typedef Types17<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
      T16, T17, T18> Tail;
};

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>
struct Types19 {
  typedef T1 Head;
  typedef Types18<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
      T16, T17, T18, T19> Tail;
};

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>
struct Types20 {
  typedef T1 Head;
  typedef Types19<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
      T16, T17, T18, T19, T20> Tail;
};

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>
struct Types21 {
  typedef T1 Head;
  typedef Types20<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
      T16, T17, T18, T19, T20, T21> Tail;
};

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>
struct Types22 {
  typedef T1 Head;
  typedef Types21<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
      T16, T17, T18, T19, T20, T21, T22> Tail;
};

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>
struct Types23 {
  typedef T1 Head;
  typedef Types22<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
      T16, T17, T18, T19, T20, T21, T22, T23> Tail;
};

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>
struct Types24 {
  typedef T1 Head;
  typedef Types23<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
      T16, T17, T18, T19, T20, T21, T22, T23, T24> Tail;
};

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>
struct Types25 {
  typedef T1 Head;
  typedef Types24<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
      T16, T17, T18, T19, T20, T21, T22, T23, T24, T25> Tail;
};

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>
struct Types26 {
  typedef T1 Head;
  typedef Types25<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> Tail;
};

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>
struct Types27 {
  typedef T1 Head;
  typedef Types26<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> Tail;
};

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>
struct Types28 {
  typedef T1 Head;
  typedef Types27<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> Tail;
};

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>
struct Types29 {
  typedef T1 Head;
  typedef Types28<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> Tail;
};

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>
struct Types30 {
  typedef T1 Head;
  typedef Types29<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> Tail;
};

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>
struct Types31 {
  typedef T1 Head;
  typedef Types30<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> Tail;
};

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>
struct Types32 {
  typedef T1 Head;
  typedef Types31<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> Tail;
};

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>
struct Types33 {
  typedef T1 Head;
  typedef Types32<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> Tail;
};

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>
struct Types34 {
  typedef T1 Head;
  typedef Types33<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> Tail;
};

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>
struct Types35 {
  typedef T1 Head;
  typedef Types34<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> Tail;
};

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>
struct Types36 {
  typedef T1 Head;
  typedef Types35<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> Tail;
};

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>
struct Types37 {
  typedef T1 Head;
  typedef Types36<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> Tail;
};

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>
struct Types38 {
  typedef T1 Head;
  typedef Types37<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> Tail;
};

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>
struct Types39 {
  typedef T1 Head;
  typedef Types38<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> Tail;
};

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>
struct Types40 {
  typedef T1 Head;
  typedef Types39<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> Tail;
};

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>
struct Types41 {
  typedef T1 Head;
  typedef Types40<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> Tail;
};

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>
struct Types42 {
  typedef T1 Head;
  typedef Types41<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> Tail;
};

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>
struct Types43 {
  typedef T1 Head;
  typedef Types42<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> Tail;
};

```

```
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>
struct Types44 {
  typedef T1 Head;
  typedef Types43<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> Tail;
};

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>
struct Types45 {
  typedef T1 Head;
  typedef Types44<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> Tail;
};

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>
struct Types46 {
  typedef T1 Head;
  typedef Types45<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> Tail;
};

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>
struct Types47 {
  typedef T1 Head;
  typedef Types46<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> Tail;
};

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>
struct Types48 {
  typedef T1 Head;
  typedef Types47<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> Tail;
};
```

```

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>
struct Types49 {
  typedef T1 Head;
  typedef Types48<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> Tail;
};

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>
struct Types50 {
  typedef T1 Head;
  typedef Types49<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> Tail;
};


}  // namespace internal

// We don’t want to require the users to write TypesN<...> directly,
// as that would require them to count the length.  Types<...> is much
// easier to write, but generates horrible messages when there is a
// compiler error, as gcc insists on printing out each template
// argument, even if it has the default value (this means Types<int>
// will appear as Types<int, None, None, ..., None> in the compiler
// errors).
//
// Our solution is to combine the best part of the two approaches: a
// user would write Types<T1, ..., TN>, and Google Test will translate
// that to TypesN<T1, ..., TN> internally to make error messages
// readable.  The translation is done by the ’type’ member of the
// Types template.
template <typename T1 = internal::None, typename T2 = internal::None,
    typename T3 = internal::None, typename T4 = internal::None,
    typename T5 = internal::None, typename T6 = internal::None,
    typename T7 = internal::None, typename T8 = internal::None,
    typename T9 = internal::None, typename T10 = internal::None,
    typename T11 = internal::None, typename T12 = internal::None,
    typename T13 = internal::None, typename T14 = internal::None,
    typename T15 = internal::None, typename T16 = internal::None,
    typename T17 = internal::None, typename T18 = internal::None,
    typename T19 = internal::None, typename T20 = internal::None,
    typename T21 = internal::None, typename T22 = internal::None,
    typename T23 = internal::None, typename T24 = internal::None,
    typename T25 = internal::None, typename T26 = internal::None,
    typename T27 = internal::None, typename T28 = internal::None,
    typename T29 = internal::None, typename T30 = internal::None,
    typename T31 = internal::None, typename T32 = internal::None,
    typename T33 = internal::None, typename T34 = internal::None,
    typename T35 = internal::None, typename T36 = internal::None,
    typename T37 = internal::None, typename T38 = internal::None,
    typename T39 = internal::None, typename T40 = internal::None,
    typename T41 = internal::None, typename T42 = internal::None,
    typename T43 = internal::None, typename T44 = internal::None,
    typename T45 = internal::None, typename T46 = internal::None,
    typename T47 = internal::None, typename T48 = internal::None,
    typename T49 = internal::None, typename T50 = internal::None>
struct Types {
  typedef internal::Types50<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> type;
};

template <>
struct Types<internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None> {
  typedef internal::Types0 type;
};
template <typename T1>
struct Types<T1, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None> {
  typedef internal::Types1<T1> type;
};
template <typename T1, typename T2>
struct Types<T1, T2, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None> {
  typedef internal::Types2<T1, T2> type;
};
template <typename T1, typename T2, typename T3>
struct Types<T1, T2, T3, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None> {
  typedef internal::Types3<T1, T2, T3> type;
};
template <typename T1, typename T2, typename T3, typename T4>
struct Types<T1, T2, T3, T4, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None> {
  typedef internal::Types4<T1, T2, T3, T4> type;
};
template <typename T1, typename T2, typename T3, typename T4, typename T5>
struct Types<T1, T2, T3, T4, T5, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None> {
  typedef internal::Types5<T1, T2, T3, T4, T5> type;
};
template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6>
struct Types<T1, T2, T3, T4, T5, T6, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None> {
  typedef internal::Types6<T1, T2, T3, T4, T5, T6> type;
};
template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7>
struct Types<T1, T2, T3, T4, T5, T6, T7, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None> {
  typedef internal::Types7<T1, T2, T3, T4, T5, T6, T7> type;
};
template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7, typename T8>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None> {
  typedef internal::Types8<T1, T2, T3, T4, T5, T6, T7, T8> type;
};
template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7, typename T8, typename T9>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None> {
  typedef internal::Types9<T1, T2, T3, T4, T5, T6, T7, T8, T9> type;
};
template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7, typename T8, typename T9, typename T10>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None> {
  typedef internal::Types10<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10> type;
};
template <typename T1, typename T2, typename T3, typename T4, typename T5,
    typename T6, typename T7, typename T8, typename T9, typename T10,
    typename T11>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None> {
  typedef internal::Types11<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11> type;
};
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>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None> {
  typedef internal::Types12<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11,
      T12> type;
};
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>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None> {
  typedef internal::Types13<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
      T13> type;
};
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>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None> {
  typedef internal::Types14<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
      T13, T14> type;
};
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>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None> {
  typedef internal::Types15<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
      T13, T14, T15> type;
};
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>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
    T16, internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None> {
  typedef internal::Types16<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
      T13, T14, T15, T16> type;
};
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>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
    T16, T17, internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None> {
  typedef internal::Types17<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
      T13, T14, T15, T16, T17> type;
};
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>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
    T16, T17, T18, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None> {
  typedef internal::Types18<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
      T13, T14, T15, T16, T17, T18> type;
};
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>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
    T16, T17, T18, T19, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None> {
  typedef internal::Types19<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
      T13, T14, T15, T16, T17, T18, T19> type;
};
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>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
    T16, T17, T18, T19, T20, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None> {
  typedef internal::Types20<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
      T13, T14, T15, T16, T17, T18, T19, T20> type;
};
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>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
    T16, T17, T18, T19, T20, T21, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None> {
  typedef internal::Types21<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
      T13, T14, T15, T16, T17, T18, T19, T20, T21> type;
};
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>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
    T16, T17, T18, T19, T20, T21, T22, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None> {
  typedef internal::Types22<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
      T13, T14, T15, T16, T17, T18, T19, T20, T21, T22> type;
};
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>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
    T16, T17, T18, T19, T20, T21, T22, T23, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None> {
  typedef internal::Types23<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
      T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23> type;
};
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>
struct Types<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15,
    T16, T17, T18, T19, T20, T21, T22, T23, T24, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None> {
  typedef internal::Types24<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
      T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24> type;
};
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>
struct Types<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, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None> {
  typedef internal::Types25<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> type;
};
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>
struct Types<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, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None> {
  typedef internal::Types26<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> type;
};
```

```
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>
struct Types<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, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None> {
  typedef internal::Types27<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> type;
};
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>
struct Types<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,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None> {
  typedef internal::Types28<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> type;
};
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>
struct Types<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,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None> {
  typedef internal::Types29<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> type;
};
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>
struct Types<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,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None> {
  typedef internal::Types30<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> type;
};
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>
struct Types<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, internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None> {
  typedef internal::Types31<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> type;
};
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>
struct Types<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, internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None> {
  typedef internal::Types32<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> type;
};
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>
struct Types<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, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None> {
  typedef internal::Types33<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> type;
};
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>
struct Types<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, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None> {
  typedef internal::Types34<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> type;
};
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>
struct Types<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, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None> {
  typedef internal::Types35<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> type;
};
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>
struct Types<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, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None> {
```

```
  typedef internal::Types36<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> type;
};
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>
struct Types<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, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None> {
  typedef internal::Types37<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> type;
};
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>
struct Types<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, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None> {
  typedef internal::Types38<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> type;
};
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>
struct Types<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, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None> {
  typedef internal::Types39<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> type;
};
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>
struct Types<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, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None> {
  typedef internal::Types40<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> type;
};
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>
struct Types<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, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None, internal::None> {
  typedef internal::Types41<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> type;
};
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>
struct Types<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, internal::None,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None> {
  typedef internal::Types42<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> type;
};
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>
struct Types<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,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None, internal::None> {
  typedef internal::Types43<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> type;
};
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>
struct Types<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,
    internal::None, internal::None, internal::None, internal::None,
    internal::None, internal::None> {
  typedef internal::Types44<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> type;
};
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>
struct Types<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,
    internal::None, internal::None, internal::None, internal::None,
    internal::None> {
  typedef internal::Types45<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> type;
};
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>
struct Types<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, internal::None, internal::None, internal::None, internal::None> {
  typedef internal::Types46<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> type;
};
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>
struct Types<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, internal::None, internal::None, internal::None> {
  typedef internal::Types47<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> type;
};
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>
struct Types<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, internal::None, internal::None> {
  typedef internal::Types48<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> type;
};
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>
struct Types<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, internal::None> {
  typedef internal::Types49<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> type;
};

namespace internal {

# define GTEST_TEMPLATE_ template <typename T> class

// The template "selector" struct TemplateSel<Tmpl> is used to
// represent Tmpl, which must be a class template with one type
// parameter, as a type.  TemplateSel<Tmpl>::Bind<T>::type is defined
// as the type Tmpl<T>.  This allows us to actually instantiate the
// template "selected" by TemplateSel<Tmpl>.
//
// This trick is necessary for simulating typedef for class templates,
// which C++ doesn't support directly.
```

```
template <GTEST_TEMPLATE_ Tmpl>
struct TemplateSel {
  template <typename T>
  struct Bind {
    typedef Tmpl<T> type;
  };
};

# define GTEST_BIND_(TmplSel, T) \
  TmplSel::template Bind<T>::type

// A unique struct template used as the default value for the
// arguments of class template Templates.  This allows us to simulate
// variadic templates (e.g. Templates<int>, Templates<int, double>,
// and etc), which C++ doesn't support directly.
template <typename T>
struct NoneT {};

// The following family of struct and struct templates are used to
// represent template lists.  In particular, TemplatesN<T1, T2, ...,
// TN> represents a list of N templates (T1, T2, ..., and TN).  Except
// for Templates0, every struct in the family has two member types:
// Head for the selector of the first template in the list, and Tail
// for the rest of the list.

// The empty template list.
struct Templates0 {};

// Template lists of length 1, 2, 3, and so on.

template <GTEST_TEMPLATE_ T1>
struct Templates1 {
  typedef TemplateSel<T1> Head;
  typedef Templates0 Tail;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2>
struct Templates2 {
  typedef TemplateSel<T1> Head;
  typedef Templates1<T2> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3>
struct Templates3 {
  typedef TemplateSel<T1> Head;
  typedef Templates2<T2, T3> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4>
struct Templates4 {
  typedef TemplateSel<T1> Head;
  typedef Templates3<T2, T3, T4> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5>
struct Templates5 {
  typedef TemplateSel<T1> Head;
  typedef Templates4<T2, T3, T4, T5> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6>
struct Templates6 {
  typedef TemplateSel<T1> Head;
  typedef Templates5<T2, T3, T4, T5, T6> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7>
struct Templates7 {
  typedef TemplateSel<T1> Head;
  typedef Templates6<T2, T3, T4, T5, T6, T7> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8>
struct Templates8 {
  typedef TemplateSel<T1> Head;
  typedef Templates7<T2, T3, T4, T5, T6, T7, T8> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9>
```

```
struct Templates9 {
  typedef TemplateSel<T1> Head;
  typedef Templates8<T2, T3, T4, T5, T6, T7, T8, T9> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10>
struct Templates10 {
  typedef TemplateSel<T1> Head;
  typedef Templates9<T2, T3, T4, T5, T6, T7, T8, T9, T10> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11>
struct Templates11 {
  typedef TemplateSel<T1> Head;
  typedef Templates10<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12>
struct Templates12 {
  typedef TemplateSel<T1> Head;
  typedef Templates11<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13>
struct Templates13 {
  typedef TemplateSel<T1> Head;
  typedef Templates12<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14>
struct Templates14 {
  typedef TemplateSel<T1> Head;
  typedef Templates13<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
      T14> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15>
struct Templates15 {
  typedef TemplateSel<T1> Head;
  typedef Templates14<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
      T15> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16>
struct Templates16 {
  typedef TemplateSel<T1> Head;
  typedef Templates15<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
      T15, T16> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17>
struct Templates17 {
  typedef TemplateSel<T1> Head;
  typedef Templates16<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
      T15, T16, T17> Tail;
};
```

```

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18>
struct Templates18 {
  typedef TemplateSel<T1> Head;
  typedef Templates17<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
      T15, T16, T17, T18> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19>
struct Templates19 {
  typedef TemplateSel<T1> Head;
  typedef Templates18<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
      T15, T16, T17, T18, T19> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20>
struct Templates20 {
  typedef TemplateSel<T1> Head;
  typedef Templates19<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
      T15, T16, T17, T18, T19, T20> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21>
struct Templates21 {
  typedef TemplateSel<T1> Head;
  typedef Templates20<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
      T15, T16, T17, T18, T19, T20, T21> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22>
struct Templates22 {
  typedef TemplateSel<T1> Head;
  typedef Templates21<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
      T15, T16, T17, T18, T19, T20, T21, T22> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23>
struct Templates23 {
  typedef TemplateSel<T1> Head;
  typedef Templates22<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
      T15, T16, T17, T18, T19, T20, T21, T22, T23> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24>
struct Templates24 {
  typedef TemplateSel<T1> Head;
  typedef Templates23<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
      T15, T16, T17, T18, T19, T20, T21, T22, T23, T24> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25>
struct Templates25 {
  typedef TemplateSel<T1> Head;
  typedef Templates24<T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
      T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26>
struct Templates26 {
  typedef TemplateSel<T1> Head;
  typedef Templates25<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27>
struct Templates27 {
  typedef TemplateSel<T1> Head;
  typedef Templates26<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28>
struct Templates28 {
  typedef TemplateSel<T1> Head;
  typedef Templates27<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29>
struct Templates29 {
  typedef TemplateSel<T1> Head;
  typedef Templates28<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30>
struct Templates30 {
  typedef TemplateSel<T1> Head;
  typedef Templates29<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31>
struct Templates31 {
  typedef TemplateSel<T1> Head;
  typedef Templates30<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32>
struct Templates32 {
  typedef TemplateSel<T1> Head;
  typedef Templates31<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33>
struct Templates33 {
  typedef TemplateSel<T1> Head;
  typedef Templates32<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34>
struct Templates34 {
  typedef TemplateSel<T1> Head;
  typedef Templates33<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35>
struct Templates35 {
  typedef TemplateSel<T1> Head;
  typedef Templates34<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36>
struct Templates36 {
  typedef TemplateSel<T1> Head;
  typedef Templates35<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37>
struct Templates37 {
  typedef TemplateSel<T1> Head;
  typedef Templates36<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38>
struct Templates38 {
  typedef TemplateSel<T1> Head;
  typedef Templates37<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39>
struct Templates39 {
  typedef TemplateSel<T1> Head;
  typedef Templates38<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40>
struct Templates40 {
  typedef TemplateSel<T1> Head;
  typedef Templates39<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41>
struct Templates41 {
  typedef TemplateSel<T1> Head;
  typedef Templates40<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42>
struct Templates42 {
  typedef TemplateSel<T1> Head;
  typedef Templates41<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
    GTEST_TEMPLATE_ T43>
struct Templates43 {
  typedef TemplateSel<T1> Head;
  typedef Templates42<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
    GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44>
struct Templates44 {
  typedef TemplateSel<T1> Head;
  typedef Templates43<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
    GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45>
struct Templates45 {
  typedef TemplateSel<T1> Head;
  typedef Templates44<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
    GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
    GTEST_TEMPLATE_ T46>
struct Templates46 {
  typedef TemplateSel<T1> Head;
  typedef Templates45<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
    GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
    GTEST_TEMPLATE_ T46, GTEST_TEMPLATE_ T47>
struct Templates47 {
  typedef TemplateSel<T1> Head;
  typedef Templates46<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
    GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
    GTEST_TEMPLATE_ T46, GTEST_TEMPLATE_ T47, GTEST_TEMPLATE_ T48>
struct Templates48 {
  typedef TemplateSel<T1> Head;
  typedef Templates47<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
    GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
    GTEST_TEMPLATE_ T46, GTEST_TEMPLATE_ T47, GTEST_TEMPLATE_ T48,
    GTEST_TEMPLATE_ T49>
struct Templates49 {
  typedef TemplateSel<T1> Head;
  typedef Templates48<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> Tail;
};

template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
    GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
    GTEST_TEMPLATE_ T46, GTEST_TEMPLATE_ T47, GTEST_TEMPLATE_ T48,
    GTEST_TEMPLATE_ T49, GTEST_TEMPLATE_ T50>
struct Templates50 {
  typedef TemplateSel<T1> Head;
  typedef Templates49<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> Tail;
};


// We don’t want to require the users to write TemplatesN<...> directly,
// as that would require them to count the length.  Templates<...> is much
// easier to write, but generates horrible messages when there is a
// compiler error, as gcc insists on printing out each template
// argument, even if it has the default value (this means Templates<list>
// will appear as Templates<list, NoneT, NoneT, ..., NoneT> in the compiler
// errors).
//
// Our solution is to combine the best part of the two approaches: a
// user would write Templates<T1, ..., TN>, and Google Test will translate
// that to TemplatesN<T1, ..., TN> internally to make error messages
// readable.  The translation is done by the ’type’ member of the
// Templates template.
template <GTEST_TEMPLATE_ T1 = NoneT, GTEST_TEMPLATE_ T2 = NoneT,
    GTEST_TEMPLATE_ T3 = NoneT, GTEST_TEMPLATE_ T4 = NoneT,
    GTEST_TEMPLATE_ T5 = NoneT, GTEST_TEMPLATE_ T6 = NoneT,
    GTEST_TEMPLATE_ T7 = NoneT, GTEST_TEMPLATE_ T8 = NoneT,
    GTEST_TEMPLATE_ T9 = NoneT, GTEST_TEMPLATE_ T10 = NoneT,
    GTEST_TEMPLATE_ T11 = NoneT, GTEST_TEMPLATE_ T12 = NoneT,
    GTEST_TEMPLATE_ T13 = NoneT, GTEST_TEMPLATE_ T14 = NoneT,
    GTEST_TEMPLATE_ T15 = NoneT, GTEST_TEMPLATE_ T16 = NoneT,
    GTEST_TEMPLATE_ T17 = NoneT, GTEST_TEMPLATE_ T18 = NoneT,
    GTEST_TEMPLATE_ T19 = NoneT, GTEST_TEMPLATE_ T20 = NoneT,
    GTEST_TEMPLATE_ T21 = NoneT, GTEST_TEMPLATE_ T22 = NoneT,
    GTEST_TEMPLATE_ T23 = NoneT, GTEST_TEMPLATE_ T24 = NoneT,
    GTEST_TEMPLATE_ T25 = NoneT, GTEST_TEMPLATE_ T26 = NoneT,
    GTEST_TEMPLATE_ T27 = NoneT, GTEST_TEMPLATE_ T28 = NoneT,
    GTEST_TEMPLATE_ T29 = NoneT, GTEST_TEMPLATE_ T30 = NoneT,
    GTEST_TEMPLATE_ T31 = NoneT, GTEST_TEMPLATE_ T32 = NoneT,
    GTEST_TEMPLATE_ T33 = NoneT, GTEST_TEMPLATE_ T34 = NoneT,
    GTEST_TEMPLATE_ T35 = NoneT, GTEST_TEMPLATE_ T36 = NoneT,
    GTEST_TEMPLATE_ T37 = NoneT, GTEST_TEMPLATE_ T38 = NoneT,
    GTEST_TEMPLATE_ T39 = NoneT, GTEST_TEMPLATE_ T40 = NoneT,
    GTEST_TEMPLATE_ T41 = NoneT, GTEST_TEMPLATE_ T42 = NoneT,
    GTEST_TEMPLATE_ T43 = NoneT, GTEST_TEMPLATE_ T44 = NoneT,
    GTEST_TEMPLATE_ T45 = NoneT, GTEST_TEMPLATE_ T46 = NoneT,
    GTEST_TEMPLATE_ T47 = NoneT, GTEST_TEMPLATE_ T48 = NoneT,
    GTEST_TEMPLATE_ T49 = NoneT, GTEST_TEMPLATE_ T50 = NoneT>
struct Templates {
  typedef Templates50<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> type;
};

template <>
struct Templates<NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT> {
  typedef Templates0 type;
};
template <GTEST_TEMPLATE_ T1>
struct Templates<T1, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT> {
  typedef Templates1<T1> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2>
struct Templates<T1, T2, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT> {
  typedef Templates2<T1, T2> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3>
struct Templates<T1, T2, T3, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates3<T1, T2, T3> type;
```

```
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4>
struct Templates<T1, T2, T3, T4, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates4<T1, T2, T3, T4> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5>
struct Templates<T1, T2, T3, T4, T5, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates5<T1, T2, T3, T4, T5> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6>
struct Templates<T1, T2, T3, T4, T5, T6, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates6<T1, T2, T3, T4, T5, T6> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7>
struct Templates<T1, T2, T3, T4, T5, T6, T7, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates7<T1, T2, T3, T4, T5, T6, T7> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates8<T1, T2, T3, T4, T5, T6, T7, T8> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates9<T1, T2, T3, T4, T5, T6, T7, T8, T9> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates10<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates11<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates12<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates13<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12,
      T13> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates14<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
      T14> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
    T15, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates15<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
      T14, T15> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
    T15, T16, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates16<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
      T14, T15, T16> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
    T15, T16, T17, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates17<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
      T14, T15, T16, T17> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
    T15, T16, T17, T18, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT> {
  typedef Templates18<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
      T14, T15, T16, T17, T18> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
    T15, T16, T17, T18, T19, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT> {
  typedef Templates19<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
      T14, T15, T16, T17, T18, T19> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
    T15, T16, T17, T18, T19, T20, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT> {
  typedef Templates20<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
      T14, T15, T16, T17, T18, T19, T20> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
    T15, T16, T17, T18, T19, T20, T21, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT> {
  typedef Templates21<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
      T14, T15, T16, T17, T18, T19, T20, T21> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
    T15, T16, T17, T18, T19, T20, T21, T22, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT> {
  typedef Templates22<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
      T14, T15, T16, T17, T18, T19, T20, T21, T22> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
    T15, T16, T17, T18, T19, T20, T21, T22, T23, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT> {
  typedef Templates23<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
      T14, T15, T16, T17, T18, T19, T20, T21, T22, T23> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24>
struct Templates<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14,
    T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT> {
  typedef Templates24<T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13,
      T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25>
struct Templates<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, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT> {
  typedef Templates25<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26>
struct Templates<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, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT> {
  typedef Templates26<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27>
struct Templates<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, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT> {
  typedef Templates27<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28>
struct Templates<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,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT> {
  typedef Templates28<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29>
struct Templates<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,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT> {
  typedef Templates29<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30>
struct Templates<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, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates30<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31>
struct Templates<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, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates31<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32>
struct Templates<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, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates32<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33>
struct Templates<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, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates33<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34>
struct Templates<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, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates34<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35>
struct Templates<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, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates35<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36>
struct Templates<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, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates36<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37>
struct Templates<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, NoneT, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates37<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38>
struct Templates<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, NoneT, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates38<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39>
struct Templates<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, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates39<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40>
struct Templates<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, NoneT, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates40<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41>
struct Templates<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, NoneT, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates41<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42>
struct Templates<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, NoneT,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates42<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
    GTEST_TEMPLATE_ T43>
struct Templates<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,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates43<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
    GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44>
struct Templates<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,
    NoneT, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates44<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
    GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45>
struct Templates<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, NoneT, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates45<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
    GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
    GTEST_TEMPLATE_ T46>
struct Templates<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, NoneT, NoneT, NoneT, NoneT> {
  typedef Templates46<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
    GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
    GTEST_TEMPLATE_ T46, GTEST_TEMPLATE_ T47>
struct Templates<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, NoneT, NoneT, NoneT> {
  typedef Templates47<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
    GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
    GTEST_TEMPLATE_ T46, GTEST_TEMPLATE_ T47, GTEST_TEMPLATE_ T48>
struct Templates<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, NoneT, NoneT> {
  typedef Templates48<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> type;
};
template <GTEST_TEMPLATE_ T1, GTEST_TEMPLATE_ T2, GTEST_TEMPLATE_ T3,
    GTEST_TEMPLATE_ T4, GTEST_TEMPLATE_ T5, GTEST_TEMPLATE_ T6,
    GTEST_TEMPLATE_ T7, GTEST_TEMPLATE_ T8, GTEST_TEMPLATE_ T9,
    GTEST_TEMPLATE_ T10, GTEST_TEMPLATE_ T11, GTEST_TEMPLATE_ T12,
    GTEST_TEMPLATE_ T13, GTEST_TEMPLATE_ T14, GTEST_TEMPLATE_ T15,
    GTEST_TEMPLATE_ T16, GTEST_TEMPLATE_ T17, GTEST_TEMPLATE_ T18,
    GTEST_TEMPLATE_ T19, GTEST_TEMPLATE_ T20, GTEST_TEMPLATE_ T21,
    GTEST_TEMPLATE_ T22, GTEST_TEMPLATE_ T23, GTEST_TEMPLATE_ T24,
    GTEST_TEMPLATE_ T25, GTEST_TEMPLATE_ T26, GTEST_TEMPLATE_ T27,
    GTEST_TEMPLATE_ T28, GTEST_TEMPLATE_ T29, GTEST_TEMPLATE_ T30,
    GTEST_TEMPLATE_ T31, GTEST_TEMPLATE_ T32, GTEST_TEMPLATE_ T33,
    GTEST_TEMPLATE_ T34, GTEST_TEMPLATE_ T35, GTEST_TEMPLATE_ T36,
    GTEST_TEMPLATE_ T37, GTEST_TEMPLATE_ T38, GTEST_TEMPLATE_ T39,
    GTEST_TEMPLATE_ T40, GTEST_TEMPLATE_ T41, GTEST_TEMPLATE_ T42,
    GTEST_TEMPLATE_ T43, GTEST_TEMPLATE_ T44, GTEST_TEMPLATE_ T45,
    GTEST_TEMPLATE_ T46, GTEST_TEMPLATE_ T47, GTEST_TEMPLATE_ T48,
    GTEST_TEMPLATE_ T49>
struct Templates<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, NoneT> {
  typedef Templates49<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> type;
};

// The TypeList template makes it possible to use either a single type
// or a Types<...> list in TYPED_TEST_CASE() and
// INSTANTIATE_TYPED_TEST_CASE_P().

template <typename T>
struct TypeList {
  typedef Types1<T> type;
};

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>
struct TypeList<Types<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> > {
  typedef typename Types<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>::type type;
};

#endif  // GTEST_HAS_TYPED_TEST || GTEST_HAS_TYPED_TEST_P

}  // namespace internal
}  // namespace testing

#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.

```
//
// pch.h
//

#pragma once

#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 )
```