C++17 NOW

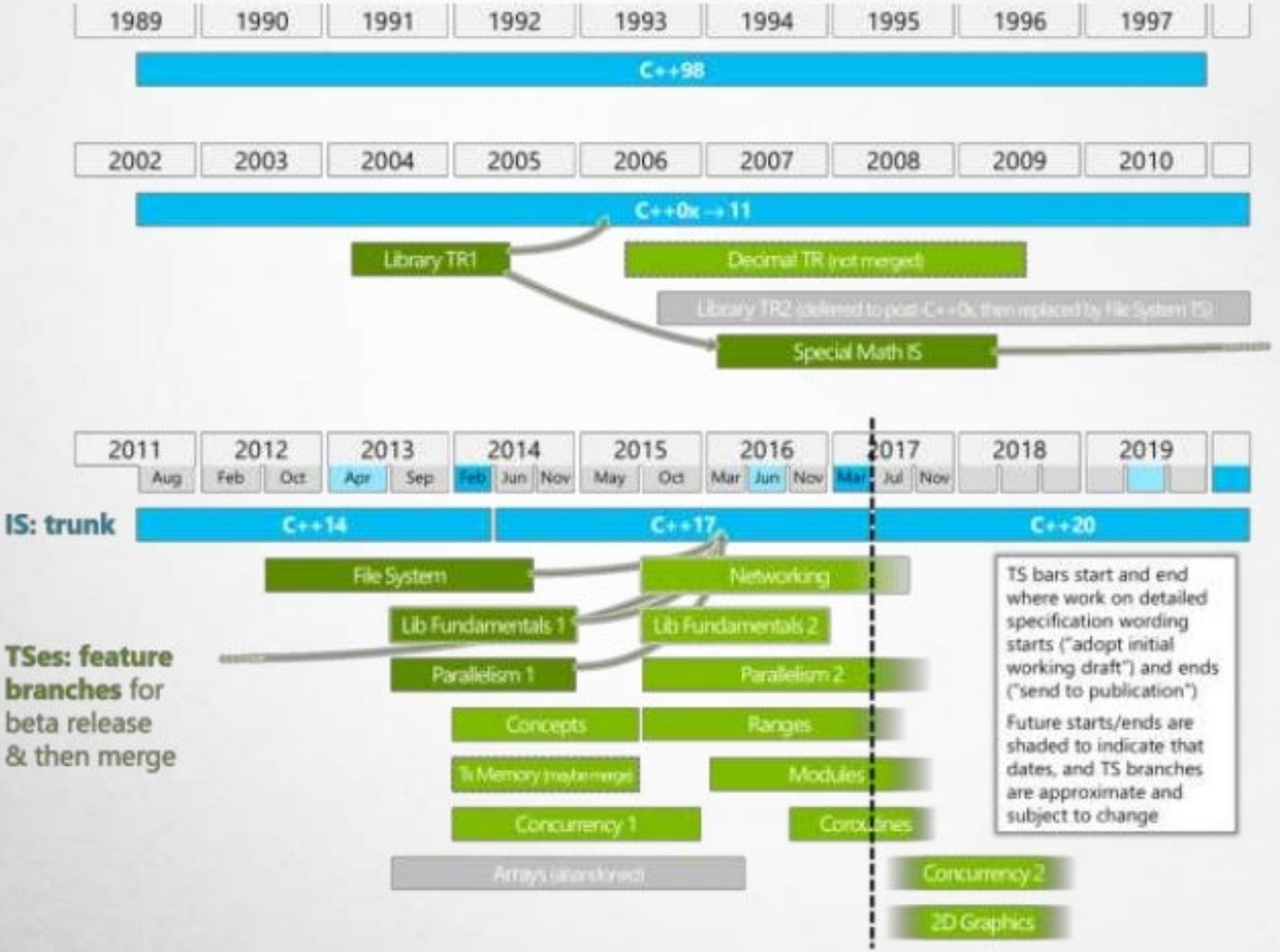
IGOR SADCHENKO

IGOR.SADCHENKO@GMAIL.COM

STANDARDIZATION

CURRENT STATUS

WARGAMING.NET
LET'S BATTLE



Standards catalogue | Publications and products

Store › Standards catalogue › Browse by ICS › 35 › 35.060 › ISO/IEC DIS 14882

ISO/IEC DIS 14882

Programming languages -- C++

General information

Current status : Under development

Edition : 5 **Number of pages :** 1617

Technical Committee : ISO/IEC JTC 1/SC 22 Programming languages, their environments and system software interfaces

ICS : 35.060 Languages used in information technology

Got a question?

Check out our FAQs

Customer care

+41 22 749 08 88
customerservice@iso.org

Opening hours:
Monday to Friday - 09:00-12:00, 14:00-17:00 (UTC+1)

Life cycle

A standard is reviewed every 5 years

00 | 10 | 20 | 30 | **40.00** Enquiry | 50 | 60 | 90 | 95

40.00 DIS registered

40.20
DIS ballot initiated: 12 weeks

40.60 Close of voting

Revisions / Corrigenda

Previously
ISO/IEC 14882:2014

Now under development
ISO/IEC DIS 14882

WARGAMING.NET
LET'S BATTLE

- WG21 C++ Russia

C++17 COMPILER SUPPORT

WARGAMING.NET
LET'S BATTLE

C++ 2017 feature	Paper(s)	Version	GCC	Clang	MSVC
New auto rules for direct-list-initialization	N3922	c++17-lang	5.0	3.8	19.0*
static_assert with no message	N3928	c++17-lang	6	2.5	19.1*
typename in a template template parameter	N4051	c++17-lang	5.0	3.5	19.0*
Removing trigraphs	N4086	c++17-lang	5.1	3.5	10.0*
Nested namespace definition	N4230	c++17-lang	6	3.6	19.0*
Attributes for namespaces and enumerators	N4266	c++17-lang	4.9 (namespaces) / 6 (enumerators)	3.6	19.0*
u8 character literals	N4267	c++17-lang	6	3.6	19.0*
Allow constant evaluation for all non-type template arguments	N4268	c++17-lang	6	3.6	
Fold Expressions	N4295	c++17-lang	6	3.6	
Remove Deprecated Use of the register Keyword	P0001R1	c++17-lang	7	3.8	
Remove Deprecated operator++(bool)	P0002R1	c++17-lang	7	3.8	
Removing Deprecated Exception Specifications from C++17	P0003R5	c++17-lang	7	4	
Make exception specifications part of the type system	P0012R1	c++17-lang	7	4	
Aggregate initialization of classes with base classes	P0017R1	c++17-lang	7	3.9	

WARGAMING.NET
LET'S BATTLE

Lambda capture of *this	P0018R3	c++17-lang	7	3.9	
Using attribute namespaces without repetition	P0028R4	c++17-lang	7	3.9	
Dynamic memory allocation for over-aligned data	P0035R4	c++17-lang	7	4	
Unary fold expressions and empty parameter packs	P0036R0	c++17-lang	6	3.9	
`__has_include` in preprocessor conditionals	P0061R1	c++17-lang	5.0	Yes	
Template argument deduction for class templates	P0091R3	c++17-lang	7	5	
Non-type template parameters with auto type	P0127R2	c++17-lang	7	4	
Guaranteed copy elision	P0135R1	c++17-lang	7	4	
New specification for inheriting constructors (DR1941 et al)	P0136R1	c++17-lang	7	3.9	
Direct-list-initialization of enumerations	P0138R2	c++17-lang	7	3.9	
Stricter expression evaluation order	P0145R3	c++17-lang	7	4	
constexpr lambda expressions	P0170R1	c++17-lang	7	5	
Differing begin and end types in range-based for	P0184R0	c++17-lang	6	3.9	19.1*
[[fallthrough]] attribute	P0188R1	c++17-lang	7	3.9	19.1*
[[nodiscard]] attribute	P0189R1	c++17-lang	7	3.9	
Pack expansions in using-declarations	P0195R2	c++17-lang	7	4	
[[maybe_unused]] attribute	P0212R1	c++17-lang	7	3.9	

WARGAMING.NET
LET'S BATTLE

C++ feature	Paper(s)	Version	GCC	Clang	MSVC
Structured Bindings	P0217R3	c++17-lang	7	4	
Hexadecimal floating-point literals	P0245R1	c++17-lang	3.0	Yes	
Ignore unknown attributes	P0283R2	c++17-lang	Yes	3.9	
constexpr if statements	P0292R2	c++17-lang	7	3.9	
init-statements for if and switch	P0305R1	c++17-lang	7	3.9	
Inline variables	P0386R2	c++17-lang	7	3.9	
DR: Matching of template template-arguments excludes compatible templates	P0522R0	c++17-lang	7	4	
std::uncaught_exceptions()	N4259	c++17	6	3.7	19.0*
Splicing Maps and Sets	P0083R3	c++17	7		
Improving std::pair and std::tuple	N4387	c++17	Yes	4	19.0*
std::shared_mutex (untimed)	N4508	c++17	6	3.7	19.0*
Elementary string conversions	P0067R5	c++17			

DETAILS

NEW AUTO RULES FOR DIRECT-LIST-INITIALIZATION(N3922)

WARGAMING.NET
LET'S BATTLE

GCC: 5.0 Clang: 3.8 MSVS: 14.0

- direct list initialization: T object {arg1, arg2, ...};
- copy list initialization: T object = {arg1, arg2, ...};

```
auto a = { 42 };   // std::initializer_list<int>
auto b { 42 };     // now int, before was std::initializer_list<int>
auto c = { 1, 2 }; // std::initializer_list<int>
auto d { 1, 2 };   // ill-formed, too many elements
auto e { 1, 2.0};   // ill-formed, different type
```

STATIC_ASSERT WITH NO MESSAGE(N3928)

WARGAMING.NET
LET'S BATTLE

GCC: 6.0 Clang: 2.5 MSVS: 15.0

- *static_assert-declaration:*
  - *static_assert (constant-expression) ;*
  - *static_assert (constant-expression, string-literal) ;*

```
static_assert(foo > bar);
...
static_assert(foo != bar);
```

TYPENAME IN A TEMPLATE TEMPLATE PARAMETER(N4051)

WARGAMING.NET
LET'S BATTLE

GCC: 5.0 Clang: 3.5 MSVS: 14.0

```
template<typename T> struct A {};
template<typename T> using B = int;
template<template<typename> class X> struct C {};
C<A> ca; // ok
C<B> cb; // ok, not a class template
template<template<typename> typename X> struct D; // was error
```

REMOVING TRIGRAPHS(N4086)

WARGAMING.NET
LET'S BATTLE

GCC: 5.1 Clang: 3.5 MSVS: 14.0

- Removes ??=, ??(, ??>, etc

```
??=include <iostream>                                  /* #           */
int main()
??<                                                    /* {           */
        char n??(??) = "Hello CoreHard_";              /* [ and ]     */
        n??(14??) = '0' + (??-0 ??' 1 ??! 2); /* ~, ^ and | */
        std::cout << n << "??/n";                      /* ??/ = \     */
        return 0;
??>
```

- But we have digraph☺

NESTED NAMESPACE DEFINITION(N4230)

WARGAMING.NET
LET'S BATTLE

GCC: 6.0 Clang: 3.6 MSVS: 14.3

- Allows to write:

```
namespace A::B::C {
    // some code
}
```

- Rather than:

```
namespace A {
    namespace B {
        namespace C {
            // some code
        }
    }
}
```

NESTED NAMESPACE DEFINITION(N4230)

WARGAMING.NET
LET'S BATTLE

GCC: 6.0 Clang: 3.6 MSVS: 14.3

- Allows to write:

```
namespace A::B::C {
    // some code
}
```

- Rather than:

```
namespace A {
    namespace B {
        namespace C {
            // some code
        }
    }
}
```

```
namespace A::B::C::D {
    namespace A::B::C::D {
        namespace A::B::C::D {
            namespace A::B::C::D {
                // some code
            }
        }
    }
}
```

ATTRIBUTES FOR NAMESPACES AND ENUMERATORS(N4266)

WARGAMING.NET
LET'S BATTLE

GCC: 4.9(ns)/6.0(enum) Clang: 3.4 MSVS: 14.0(no enum)

```
enum E {
    foo = 0,
    foobar[[deprecated]] = foo
};
E e = foobar; // warning

namespace[[deprecated]] legacy{
    void func() {}
}
legacy::func(); // warning, error in msvs14.3)
```

U8 CHARACTER LITERALS(N4267)

WARGAMING.NET
LET'S BATTLE

GCC: 6.0 Clang: 3.6 MSVS: 14.0

- UTF-8 character literal, e.g. u8'a'. Such literal has type char and the value equal to ISO 10646 code point value of c-char, provided that the code point value is representable with a single UTF-8 code unit. If c-char is not in Basic Latin or C0 Controls Unicode block, the program is ill-formed.
- The compiler will report errors if character cannot fit inside u8 ASCII range.

```
char c = u8'Ë'; // too many characters in constant
```

ALLOW CONSTANT EVALUATION FOR ALL NON-TYPE TEMPLATE ARGUMENTS(N4268)

WARGAMING.NET
LET'S BATTLE

GCC: 6.0 Clang: 3.6 MSVS:no

```
template<int *p> struct A {};
int n;
A<&n> a; // ok

constexpr int *p() { return &n; }
A<p()> b; // error before C++17
```

WARGAMING.NET
LET'S BATTLE

FOLD EXPRESSIONS(N4295)

GCC: 6.0 Clang: 3.6 MSVS: no

- Reduces (folds) a parameter pack over a binary operator.

```
(pack op ...)          // unary right fold
(... op pack)          // unary left fold
(pack op ... op init) // binary right fold
(init op ... op pack) // binary left fold
```

- Explanation:

1) Unary right fold *(E op ...)* becomes *E1 op (... op (EN-1 op EN))*

2) Unary left fold *(... op E)* becomes *((E1 op E2) op ...) op EN*

3) Binary right fold *(E op ... op I)* becomes *E1 op (... op (EN-1 op (EN op I)))*

4) Binary left fold *(I op ... op E)* becomes *(((I op E1) op E2) op ...) op EN*

- C++17 support 32 binary operators in fold expressions

FOLD EXPRESSIONS(N4295)

WARGAMING.NET
LET'S BATTLE

GCC: 6.0 Clang: 3.6 MSVS: no

- Allows to write compact code with variadic templates without using explicit recursion. C++17 support 32 binary operators in fold expressions.
- Before:

```
bool all() {
        return true;
}

template<typename T, typename ...Args>
bool all(T t, Args ... args) {
        return t && all(args...);
}
```

- Now:

```
template<typename... Args>
bool all(Args... args) {
        return (... && args);
}
```

UNARY FOLD EXPRESSIONS AND EMPTY PARAMETER PACKS (P0036R0)

WARGAMING.NET LET'S BATTLE

GCC: 6.0 Clang: 3.9 MSVS: no

- When a unary fold is used with a pack expansion of length zero, only the following operators are allowed:
  - 1) Logical *AND* (**&&**). The value for the empty pack is true
  - 2) Logical *OR* (**||**). The value for the empty pack is false
  - 3) The *comma operator* (**,**). The value for the empty pack is void()
- For any operator not listed above, an unary fold expression with an empty parameter pack is ill-formed.

REMOVING

WARGAMING.NET
LET'S BATTLE

- Remove deprecated use of the ***register*** keyword(P0001R1)
  - GCC: 7.0 Clang: 3.8 MSVS: no
- Remove deprecated ***operator++(bool)***(P0002R1)
  - GCC: 7.0 Clang: 3.8 MSVS: no
- Removing deprecated exception specifications from C++17(P0003R5)
  - GCC: 7.0 Clang: 4.0 MSVS: no
- Removal of some deprecated types and functions, including *std::auto_ptr*, *std::random_shuffle*, and old function adaptors(N4190)

MAKE EXCEPTION SPECIFICATIONS PART OF THE TYPE SYSTEM(P0012R1)

WARGAMING.NET LET'S BATTLE

GCC: 7.0 Clang: 4.0 MSVS: no

- Until C++17 exception specifications for a function didn't belong to the type of the function, but it will be part of it.

```
void(*pf)();
void(**ppf)() noexcept = &pf;    // error: cannot convert to pointer noexcept function

struct S { typedef void(*pf)(); operator pf(); };
void(*q)() noexcept = S();    // error: cannot convert to pointer noexcept function
```

AGGREGATE INITIALIZATION OF CLASSES WITH BASE CLASSES(P0017R1)

WARGAMING.NET
LET'S BATTLE

GCC: 7.0 Clang: 3.9 MSVS: no

› *An aggregate is an array or a class with:*
** no user-provided constructors (including those inherited from a base class),*
** no private or protected non-static data members,*
** no virtual functions, and*
** no virtual, private or protected base classes*

```
struct Base { int a1, a2; };
struct Derived : Base { int b1; };

Derived d1{ { 1, 2 }, 3 };      // full explicit initialization
Derived d2{ {}, 1 };           // the base is value initialized
```

LAMBDA CAPTURE OF *THIS(P0018R3)

WARGAMING.NET
LET'S BATTLE

GCC: 7.0 Clang: 3.9 MSVS: no

- Capturing this in a lambda's environment was previously reference-only.
- *this (C++17) will now make a copy of the current object, while this (C++11) continues to capture by reference.

```
struct Foo {
     int value{ 25 };
     void test() {
          auto getValueCopy = [*this] { return value; };
          auto getValueRef = [this] { return value; };
          value = 27
          getValueCopy(); // 25
          getValueRef(); // 27
     }
};
```

__HAS_INCLUDE IN PREPROCESSOR CONDITIONALS(P0061R1)

WARGAMING.NET
LET'S BATTLE

GCC: 5.0 Clang: 5.0 MSVS: no

- This feature allows a C++ program to directly, reliably and portably determine whether or not a library header is available for inclusion.

```
#if __has_include(<string_view>)
     #include <string_view>
     #define has_string_view 1
#elif __has_include(<experimental/string_view>)
     #include <experimental/string_view>
     #define has_string_view 1
     #define experimental_string_view 1
#else
     #define has_string_view 0
#endif
```

TEMPLATE ARGUMENT DEDUCTION FOR CLASS TEMPLATES(P0091R3)

WARGAMING.NET
LET'S BATTLE

GCC: 7.0 Clang: no MSVS: no

- Before C++17, template deduction worked for functions but not for classes.
  - `std::make_pair("aaa"s, 111);`
- In C++17 template class constructors can deduce type parameters.
  - `std::pair("aaa"s, 111);`

TEMPLATE ARGUMENT DEDUCTION FOR CLASS TEMPLATES(P0091R3)

WARGAMING.NET
LET'S BATTLE

GCC: 7.0 Clang: no MSVS: no

- Before C++17, template deduction worked for functions but not for classes.
  - `std::make_pair("aaa"s, 111);`
- In C++17 template class constructors can deduce type parameters.
  - `std::pair("aaa"s, 111);`

```
std::tuple(
    [](){}, [](){}, [](){}, [](){}, [](){}, [](){},
    [](){}, [](){}, [](){}, [](){}, [](){}, [](){},
    [](){}, [](){}, [](){}, [](){}, [](){}, [](){},
);
```

NON-TYPE TEMPLATE PARAMETERS WITH AUTO TYPE(P0127R2)

WARGAMING.NET
LET'S BATTLE

GCC: 7.0 Clang: 4.0 MSVS: no

- Automatically deduce type on non-type template parameters.

```
template <auto value> void f() { }
f<10>();                        // deduces int
```

WARGAMING.NET
LET'S BATTLE

CONSTEXPR LAMBDA EXPRESSIONS(P0170R1)

GCC: 7.0 Clang: no MSVS: no

- Compile-time lambdas using *constexpr*.

```
auto identity = [](int n) constexpr { return n; };
static_assert(identity(123) == 123);

constexpr auto add = [](int x, int y) {
    auto L = [=] { return x; };
    auto R = [=] { return y; };
    return [=] { return L() + R(); };
};
static_assert(add(1, 2)() == 3);

constexpr int addOne(int n) {
    return [n] { return n + 1; }();
}
static_assert(addOne(1) == 2);
```

ATTRIBUTES

WARGAMING.NET
LET'S BATTLE

GCC: 7.0 Clang: 3.9 MSVS: no

- [[fallthrough]] attribute (P0188R1)
- [[nodiscard]] attribute(P0189R1)
- [[maybe_unused]] attribute
- Ignore unknown attributes(P0283R2)
- Using attribute namespaces without repetition(P0028R4)

PACK EXPANSIONS IN USING-DECLARATIONS(P0195R2)

WARGAMING.NET
LET'S BATTLE

GCC: 7.0 Clang: 4.0 MSVS: no

- Allows you to inject names with *using-declarations* from all types in a parameter pack.

```
template <typename T, typename... Ts>
struct Overloader : T, Overloader<Ts...> {
    using T::operator();
    using Overloader<Ts...>::operator();
    // […] until
};

template <typename T> struct Overloader<T> : T {
    using T::operator();
};
```

```
template <typename... Ts>
struct Overloader : Ts... {
    using Ts::operator()...;
    // […] C++17
};
```

DECOMPOSITION DECLARATIONS(P0217R3)

WARGAMING.NET
LET'S BATTLE

GCC: 7.0 Clang: 4.0 MSVS: no

- For example:
  - `std::tie(a, b, c) = tuple; // a, b, c need to be declared first`
  - `std::array<T, 3> f();`
    `std::tie(x, y, z) = f(); // INVALID.`
- Now we can write:
  - `auto [a, b, c] = tuple;`
  - `auto [a, b, c] = f();`

- Such expressions also work on structs, pairs, and arrays.

INIT-STATEMENTS FOR IF AND SWITCH(P0305R1)

WARGAMING.NET
LET'S BATTLE

GCC: 7.0 Clang: 3.9 MSVS: no

- New versions of the if and switch statements for C++:
  - if (init; condition) and switch (init; condition).
- Now you can write:

```
if (auto val = GetValue(); condition(val))
      // on success
else
      // on false...
```

INLINE VARIABLES(P0386R2)

WARGAMING.NET
LET'S BATTLE

GCC: 7.0 Clang: 3.9 MSVS: no

- A variable declared inline has the same semantics as a function declared inline: it can be defined, identically, in multiple translation units, must be defined in every translation unit in which it is used, and the behavior of the program is as if there is exactly one variable.

```
struct Foo
{
        static const int sValue;
};
inline int const Foo::sValue = 777;
```

- Or:

```
struct Foo
{
        inline static const int sValue = 777;
};
```

STD::UNCAUGHT_EXCEPTIONS() (N4259)

WARGAMING.NET
LET'S BATTLE

GCC: 6.0 Clang: 3.7 MSVS: 14.0

- The function returns the number of uncaught exception objects in the current thread.
- A type that wants to know whether its destructor is being run to unwind this object can query uncaught_exceptions in its constructor and store the result, then query uncaught_exceptions again in its destructor; if the result is different,
  then this destructor is being invoked as part of stack unwinding due to a new exception that was thrown later than the object's construction

CONSTEXPR IF-STATEMENTS(P0292R2)

WARGAMING.NET
LET'S BATTLE

GCC: 7.0 Clang: 3.9 MSVS: no

```
if constexpr(cond)
    statement1; // Discarded if cond is false
else
    statement2; // Discarded if cond is true
```

STD::VARIANT

WARGAMING.NET
LET'S BATTLE

- The class template *std::variant* represents a type-safe union. An instance of *std::variant* at any given time holds a value of one of its alternative types (it's also possible for it to be valueless).

```
std::variant<int, double> v{ 12 };
std::get<int>(v); // == 12
std::get<0>(v); // == 12
v = 12.0;
std::get<double>(v); // == 12.0
std::get<1>(v); // == 12.0
```

STD::OPTIONAL

WARGAMING.NET
LET'S BATTLE

› The class template std::optional manages an optional contained value, i.e. a value that may or may not be present. A common use case for optional is the return value of a function that may fail.

```
std::optional<std::string> create(bool b) {
    if (b)
        return "Godzilla";
    else
        return {};
}
create(false).value_or("empty"); // == "empty"
create(true).value(); // == "Godzilla"
// optional-returning factory functions are usable as conditions of while and if
if (auto str = create(true)) {
    // ...
}
```

STRING_VIEW

WARGAMING.NET
LET'S BATTLE

- The class template basic_string_view describes an object that can refer to a constant contiguous sequence of char-like objects with the first element of the sequence at position zero.
- A typical implementation holds only two members: a pointer to constant CharT and a size.

```
// Regular strings.
std::string_view cppstr{ "foo" };
// Wide strings.
std::wstring_view wcstr_v{ L"baz" };
// Character arrays.
char array[3] = { 'b', 'a', 'r' };
std::string_view array_v(array, sizeof array);

std::string str{ "   trim me" };
std::string_view v{ str };
v.remove_prefix(std::min(v.find_first_not_of(" "), v.size()));
str; //  == "   trim me"
v; // == "trim me"
```

STRING_VIEW

WARGAMING.NET
LET'S BATTLE

- Problem

```
class foo
{
    std::string str_;

public:
    std::string_view get_str() const
    {
        // substr starting at index 1 till the end
        return str_.substr(1u);
    }
};
```

STRING_VIEW

WARGAMING.NET
LET'S BATTLE

- Solution

```
class foo
{
    std::string str_;

public:
    std::string_view get_str() const
    {
        // substr starting at index 1 till the end
        return std::string_view(str_).substr(1u);
    }
};
```

FILE_SYSTEM

WARGAMING.NET
LET'S BATTLE

GCC: 6.0 Clang: 3.9 MSVS: 14.0

- The Filesystem library provides facilities for performing operations on file systems and their components, such as paths, regular files, and directories.
- The filesystem library was originally developed as boost.filesystem
- Structure:
  - path - represents a path
  - filesystem_error - an exception thrown on file system errors
  - directory_entry - a directory entry
  - directory_iterator - an iterator to the contents of the directory
  - recursive_directory_iterator - an iterator to the contents of a directory and its subdirectories
  - file_status - represents file type and permissions
  - space_info - information about free and available space on the filesystem
  - file_type - the type of a file

ETC

- std::any
- std::byte
- std:invoke
- std::is_callable
- std::apply
- parallels algorithm
- improved math

SUMMARY

CONCLUSION

WARGAMING.NET
LET'S BATTLE

- New standards are almost supported by compilers by the time of release
- You can help make new C++ better
- C++ gets harder
- It is C++!

LINKS

WARGAMING.NET
LET'S BATTLE

- Last draft
  - http://open-std.org/JTC1/SC22/WG21/docs/papers/2016/n4618.pdf
- Current status of C++ standardization
  - https://isocpp.org/std/status
  - https://www.iso.org/standard/68564.html
- WG21 C++ Russia
  - https://stdcpp.ru/
- C++17 compiler support
  - http://en.cppreference.com/w/cpp/compiler_support
  - https://gcc.gnu.org/projects/cxx-status.html#cxx1z
  - http://clang.llvm.org/cxx_status.html#cxx17
  - https://blogs.msdn.microsoft.com/vcblog/2017/05/10/c17-features-in-vs-2017-3/
- C++17 Features overview
  - http://www.bfilipek.com/2017/01/cpp17features.html
  - https://github.com/AnthonyCalandra/modern-cpp-features#stdstring_view

THANK YOU FOR YOUR ATTENTION!

ANY QUESTIONS?

IGOR SADCHENKO

software developer

igor.sadchenko@gmail.com

+375 33 642 92 91

https://www.facebook.com/WargamingMinsk

https://www.linkedin.com/company/wargaming-net

wargaming.com