Pre- C++ Standards Meeting

May 2015

Lenexa, Kansas City

René Meusel, Axel Naumann - PH-SFT

29th of April

Agenda

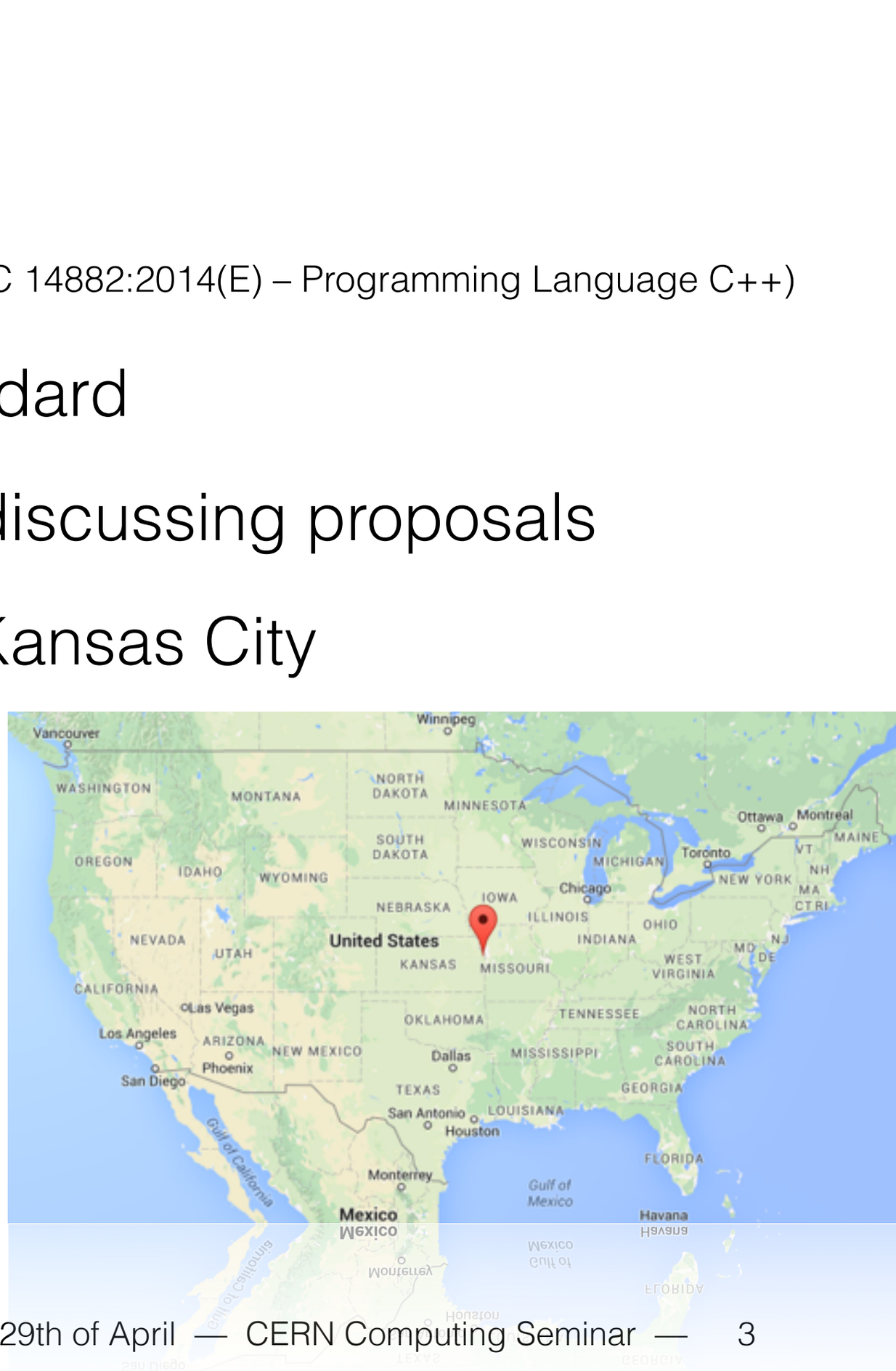
1. C++ Standard | Working groups
2. What's New in C++14
3. Interesting Proposals for Kansas (and C++17)
4. Discussion

C++ Standard

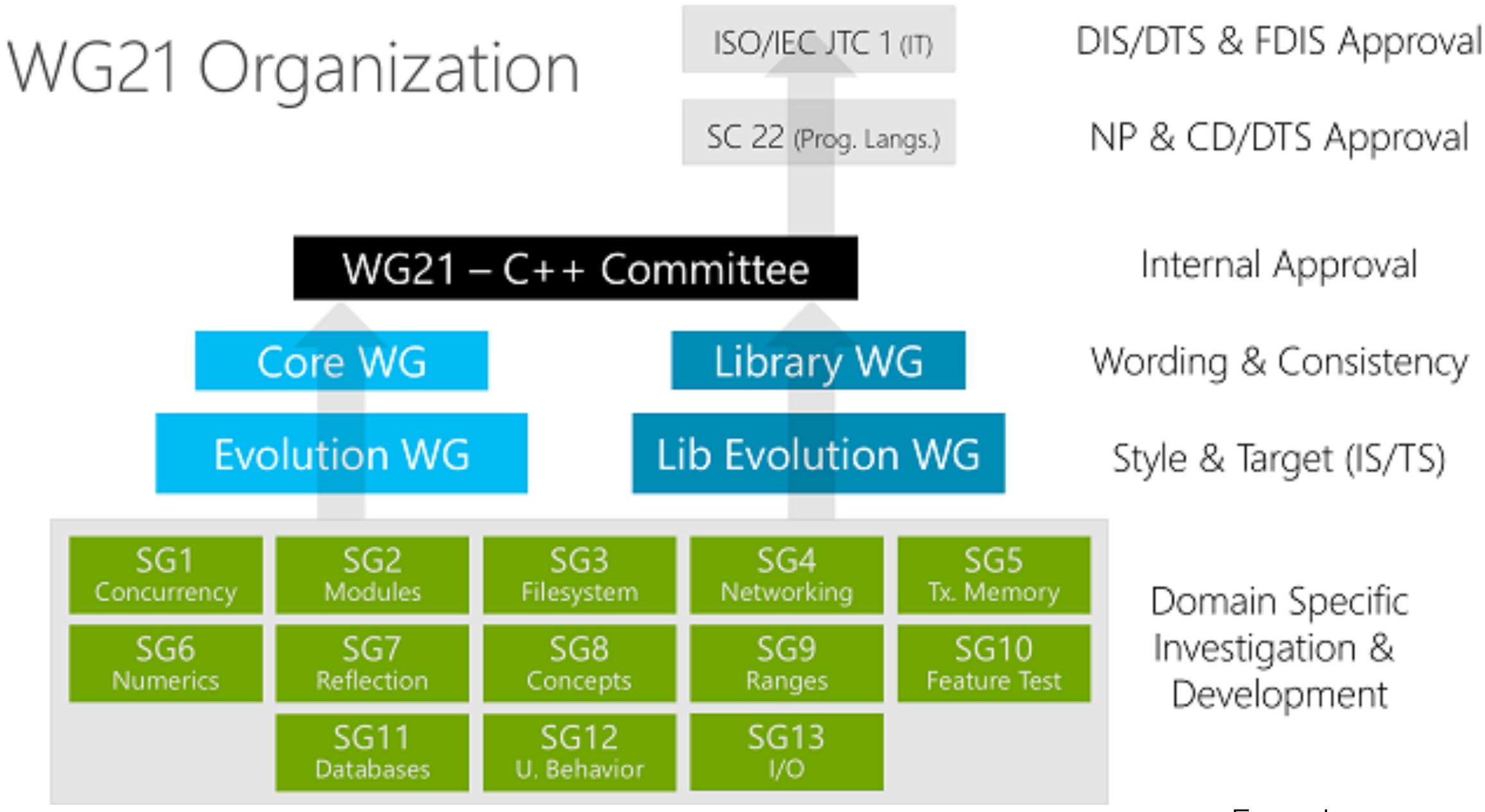
- C++ is an ISO standard (ISO/IEC 14882:2014(E) – Programming Language C++)
  - STL is included in the standard
- Meetings: 90..120 people discussing proposals
  - Next: 4th to 9th of May in Kansas City

- Axel is going!

www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/

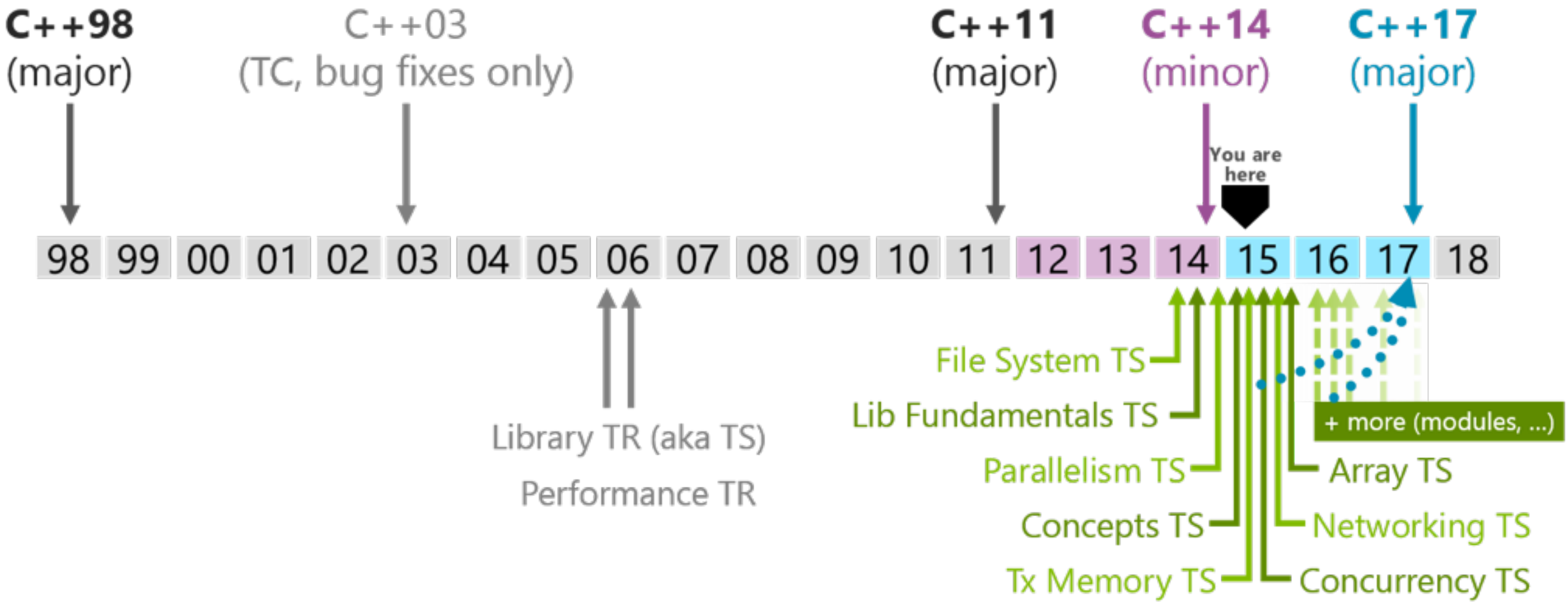


C++ Committee



From: isocpp.org

C++ Status



From: isocpp.org

C++14 Candy

```
#include <iostream>
#include <string>

template<class T>
T pi = T(3.141’592'653'589'793'238'462'643'383'279);


[[deprecated]] auto circle_area =
  [](auto r) -> auto { return pi<decltype(r)> * r * r; };

using namespace string_literals;
int main() {
  std::cout.precision(25);
  std::cout << ("Precise "s + "Circles"s)      << std::endl;
  std::cout << "Int:    " << circle_area(2)    << std::endl;
  std::cout << "Float:  " << circle_area(2.f) << std::endl;
  std::cout << "Double: " << circle_area(2.)   << std::endl;
  return 0;
}
```

C++14 Candy

```
#include <iostream>
#include <string>

template<class T>
T pi = T(3.141'592'653'589'793'238'462'643'383'279);


[[deprecated]] auto circle_area =
  [](auto r) -> auto { return pi<decltype(r)> * r * r; };
```

templated variable

```
using namespace string_literals;
int main() {
  std::cout.precision(25);
  std::cout << ("Precise "s + "Circles"s)      << std::endl;
  std::cout << "Int:    " << circle_area(2)    << std::endl;
  std::cout << "Float:  " << circle_area(2.f) << std::endl;
  std::cout << "Double: " << circle_area(2.)   << std::endl;
  return 0;
}
```

C++14 Candy

```
#include <iostream>
#include <string>

template<class T>
T pi = T(3.141’592'653'589'793'238'462'643'383'279);

[[deprecated]] auto circle_area =
  [](auto r) -> auto { return pi<decltype(r)> * r * r; };
```

digit separators

```
using namespace string_literals;
int main() {
  std::cout.precision(25);
  std::cout << ("Precise "s + "Circles"s)      << std::endl;
  std::cout << "Int:    " << circle_area(2)    << std::endl;
  std::cout << "Float:  " << circle_area(2.f) << std::endl;
  std::cout << "Double: " << circle_area(2.)   << std::endl;
  return 0;
}
```

C++14 Candy

```
#include <iostream>
#include <string>

template<class T>
T pi = T(3.141’592'653'589'793'238'462'643'383'279);
```

deprecation warning

```
[[deprecated]] auto circle_area =
  [](auto r) -> auto { return pi<decltype(r)> * r * r; };

using namespace string_literals;
int main() {
  std::cout.precision(25);
  std::cout << ("Precise "s + "Circles"s)      << std::endl;
  std::cout << "Int:    " << circle_area(2)    << std::endl;
  std::cout << "Float:  " << circle_area(2.f) << std::endl;
  std::cout << "Double: " << circle_area(2.)   << std::endl;
  return 0;
}
```

C++14 Candy

```
#include <iostream>
#include <string>

template<class T>
T pi = T(3.141’592'653'589'793'238'462'643'383'279);


[[deprecated]] auto circle_area =
  [](auto r) -> auto { return pi<decltype(r)> * r * r; };
```

generic lambdas

```
using namespace string_literals;
int main() {
  std::cout.precision(25);
  std::cout << ("Precise "s + "Circles"s)      << std::endl;
  std::cout << "Int:    " << circle_area(2)    << std::endl;
  std::cout << "Float:  " << circle_area(2.f) << std::endl;
  std::cout << "Double: " << circle_area(2.)   << std::endl;
  return 0;
}
```

C++14 Candy

```
#include <iostream>
#include <string>

template<class T>
T pi = T(3.141’592'653'589'793'238'462'643'383'279);


[[deprecated]] auto circle_area =
  [](auto r) -> auto { return pi<decltype(r)> * r * r; };

using namespace string_literals;
int main() {
  std::cout.precision(25);
  std::cout << ("Precise "s + "Circles"s)      << std::endl;
  std::cout << "Int:    " << circle_area(2)    << std::endl;
  std::cout << "Float:  " << circle_area(2.f) << std::endl;
  std::cout << "Double: " << circle_area(2.)   << std::endl;
  return 0;
}
```

library defined literals

C++14 Candy

```
#include <iostream>
#include <string>

template<class T>
T pi = T(3.141’592'653'589'793'238'462'643'383'279);


[[deprecated]] auto circle_area =
  [](auto r) -> auto { return pi<decltype(r)> * r * r; };

using namespace string_literals;
int main() {
  std::cout.precision(25);
  std::cout << "Circles"s + "Circles"s      << std::endl;
  std::cout << "Int:    " << circle_area(2)   << std::endl;
  std::cout << "Float:  " << circle_area(2.f) << std::endl;
  std::cout << "Double: " << circle_area(2.)  << std::endl;
  return 0;
}
```

only one explicit type!

C++14 Candy

```
$$ g++-5 -std=gnu++14 spike.cc
```

C++14 Candy

```
$$ g++-5 -std=gnu++14 spike.cc
spike.cc:5:31: warning: 'depricated' attribute directive
ignored [-Wattributes]
 [[depricated]] auto circle_area = [](auto r) { return
pi<decltype(r)> * r * r; };
```

C++14 Candy

```
#include <iostream>
#include <string>

template<class T>
T pi = T(3.1415926535897932385L);

[[depricated]] auto circle_area = [](auto r) { return pi<decltype(r)> * r * r; };

int main() {
  std::cout << "Precise Circles\n";
  std::cout << "Int:    " << circle_area(2)   << std::endl;
  std::cout << "Float:  " << circle_area(2.f) << std::endl;
  std::cout << "Double: " << circle_area(2.)  << std::endl;
  return 0;
}
```

```
$$ g++-5 -std=gnu++14 spike.cc
spike.cc:5:31: warning: 'depricated' attribute directive
ignored [-Wattributes]
 [[depricated]] auto circle_area = [](auto r) { return
pi<decltype(r)> * r * r; };

$$ ./spike
Precise Circles
Int:    12
Float:  12.566370964050292968 75
Double: 12.566370614359172463 99185
```

C++ Status

- https://isocpp.org/std/status
- **C++14** approved last August / published in December
  - mainly bug fixes and minor improvements of C++11
  - Function return type deduction

```
auto foo() { return 1; }
```

- Relaxed restrictions for `constexpr` functions
- Initialised lambda captures

```
int x = 4
auto y = [&z = x, x = x + 1]() -> int {
    r += 2;
    return x + 2;
}
```

Proposals for Lenexa

Concepts

- Concepts Lite

```
template <Sortable C>      // need: random access in C
void sort(C &container);   //       and "<" for elements

// alternative notation (using `requires` keyword)
template <typename C> requires Sortable<C>
void sort(C &container);

// define a simple concept for equality comparable objects
template <typename T>
concept bool EqualityComparable = requires (T a, T b) {
   { a == b } -> bool;
};

// check before method usage
template <Object T>
class vector {
   bool operator==(const vector<T>& x) const
      requires EqualityComparable<T>;
};
```

www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4377.pdf

(Tutorial: http://en.cppreference.com/w/cpp/language/constraints)

Concepts

- Concepts Lite

```
template <Sortable C>      // need: random access in C
void sort(C &container);   //       and "<" for elements
```

```
// alternative notation (using `requires` keyword)
template <typename C> requires Sortable<C>
void sort(C &container);
```

```
// define a simple concept for equality comparable objects
template <typename T>
concept bool EqualityComparable = requires (T a, T b) {
   { a == b } -> bool;
};

// check before method usage
template <Object T>
class vector {
   bool operator==(const vector<T>& x) const
      requires EqualityComparable<T>;

};
```

www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4377.pdf

(Tutorial: http://en.cppreference.com/w/cpp/language/constraints)

Concepts

- Concepts Lite

```
template <Sortable C>      // need: random access in C
void sort(C &container);   //       and "<" for elements
```

```
$$ g++ -std=gnu++17 concepts_demo.cpp
```

```
// alternative notation (using 'requires' keyword)
template <typename C> requires Sortable<C>
void sort(C &container);

// define a simple concept for equality comparable objects
template <typename T>
concept bool EqualityComparable = requires (T a, T b) {
  { a == b } -> bool;
};

// check before method usage
template <Object T>
class vector {
  bool operator==(const vector<T>& x) const
    requires EqualityComparable<T>;
};
```

www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4377.pdf
(Tutorial: http://en.cppreference.com/w/cpp/language/constraints)

Concepts

- Concepts Lite

```
template <Sortable C>      // need: random access in C
void sort(C &container);   //       and "<" for elements
```

```
$$ g++ -std=gnu++17 concepts_demo.cpp
error: no matching function for call to ‘sort(list<int>&)’
   sort(l);
            ^
  note: candidate is:
  note: template<Sortable T> void sort(T)
void sort(T t) { }
          ^
  note: template constraints not satisfied because
  note:   ‘T’ is not a/an ‘Sortable’ type [with T = list<int>]
since
  note:     ‘declval<T>()[n]’ is not valid syntax
```

www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4377.pdf

(Tutorial: http://en.cppreference.com/w/cpp/language/constraints)

```
In file included from /usr/include/c++/5/algorithm:62:0,
                 from spike.cc:4:
/usr/include/c++/5/bits/stl_algo.h: In instantiation of 'void std::__sort(_RandomAccessIterator, _RandomAccessIterator, _Compare) [with
_RandomAccessIterator = std::_List_iterator<int>; _Compare = __gnu_cxx::__ops::_Iter_less_iter]':
/usr/include/c++/5/bits/stl_algo.h:4698:18:   required from 'void std::sort(_RAIter, _RAIter) [with _RAIter = std::_List_iterator<int>'
spike.cc:9:12:   required from 'void sort(C&) [with C = std::list<int>]'
spike.cc:16:9:   required from here
/usr/include/c++/5/bits/stl_algo.h:1964:22: error: no match for 'operator-' (operand types are 'std::_List_iterator<int>' and
'std::_List_iterator<int>')
     std::__lg(__last - __first) * 2,
                      ^
In file included from /usr/include/c++/5/bits/stl_algobase.h:67:0,
                 from /usr/include/c++/5/bits/char_traits.h:39,
                 from /usr/include/c++/5/ios:40,
                 from /usr/include/c++/5/ostream:38,
                 from /usr/include/c++/5/iostream:39,
                 from spike.cc:1:
/usr/include/c++/5/bits/stl_iterator.h:328:5: note: candidate: template<class _Iterator> typename
std::reverse_iterator<_Iterator>::difference_type std::operator-(const std::reverse_iterator<_Iterator>&, const
std::reverse_iterator<_Iterator>&)
     operator-(const reverse_iterator<_Iterator>& __x,
     ^
/usr/include/c++/5/bits/stl_iterator.h:328:5: note:   template argument deduction/substitution failed:
In file included from /usr/include/c++/5/algorithm:62:0,
                 from spike.cc:4:
/usr/include/c++/5/bits/stl_algo.h:1964:22: note:   'std::_List_iterator<int>' is not derived from 'const std::reverse_iterator<_Iterator>'
     std::__lg(__last - __first) * 2,
                      ^
In file included from /usr/include/c++/5/bits/stl_algobase.h:67:0,
                 from /usr/include/c++/5/bits/char_traits.h:39,
                 from /usr/include/c++/5/ios:40,
                 from /usr/include/c++/5/ostream:38,
                 from /usr/include/c++/5/iostream:39,
                 from spike.cc:1:
/usr/include/c++/5/bits/stl_iterator.h:385:5: note: candidate: template<class _IteratorL, class _IteratorR> typename
std::reverse_iterator<_Iterator>::difference_type std::operator-(const std::reverse_iterator<_Iterator>&, const
std::reverse_iterator<_IteratorR>&)
     operator-(const reverse_iterator<_IteratorL>& __x,
     ^
/usr/include/c++/5/bits/stl_iterator.h:385:5: note:   template argument deduction/substitution failed:
In file included from /usr/include/c++/5/algorithm:62:0,
                 from spike.cc:4:
/usr/include/c++/5/bits/stl_algo.h:1964:22: note:   'std::_List_iterator<int>' is not derived from 'const std::reverse_iterator<_Iterator>'
     std::__lg(__last - __first) * 2,
                      ^
In file included from /usr/include/c++/5/vector:65:0,
                 from spike.cc:3:
/usr/include/c++/5/bits/stl_bvector.h:208:3: note: candidate: std::ptrdiff_t std::operator-(const std::_Bit_iterator_base&, const
std::_Bit_iterator_base&)
   operator-(const _Bit_iterator_base& __x, const _Bit_iterator_base& __y)
   ^
/usr/include/c++/5/bits/stl_bvector.h:208:3: note:   no known conversion for argument 1 from 'std::_List_iterator<int>' to 'const
std::_Bit_iterator_base&'
```

Library Evolution 1/3

- Propagate Const Wrapper

```
// call const-override of m_ptr->foo() from const-members
class Bar {
  void bar() const { m_ptr->foo(); }
  std::propagate_const<Foo*> m_ptr;
}
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4388.html

- User-Releasable promise<>

```
std::promise<int> p;
p.get_future();
p.set_value(42);
// p's shared state no longer needed, stays until ~promise()

std::promise<int>{}.set_value_at_thread_exit(42);
// shared state no longer needed, stays until thread exit
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4445.html

Library Evolution 2/3

- Improved Rounding and Overflow Control
  http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4448.html
- Message Digest Library

```
std::hashlib::sha1 h;
h.update("hello world");
h.digest();
```

  http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4449.html
- Networking Library *(adaption from Boost ASIO)*

```
std::vector<char> buf(1024);
for (;;) {
   std::size_t length = socket.read_some(buffer(buf));
   uppercase(buf.begin(), buf.begin() + length);
   write(socket, buffer(buf, length));
}
```

  http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4478.html

Library Evolution 3/3

- Variant: a Typesafe Union *(from Axel Naumann - CERN)*

```
variant<int, float> v; int i;
v = 12;
i = get<int>(v);
i = get<0>(v);   // same effect as the previous line
try { get<float>(v); } catch (bad_variant_access&) {}
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4450.pdf

- Atomic Views

```
void foo(int *i, int n) {
   atomic_array_view<int> ai(i, n);
   ai[0].atomic_operation();
   auto x = ai[12]; x.atomic_operation();
}
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4373.pdf

- Constant Views: `std::as_const<>`

```
std::string str = "Hello World!";
const std::string &cs = std::as_const(str);
foo(str);                  // call foo(std::string &bar)
foo(std::as_const(str); // call foo(const std::string &bar)
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4380.html

Concurrency

- Latches and Barriers

```
std::latch syn(10);
/* other threads... */ syn.arrive(); // non-blocking; --syn
syn.wait();                          // block until syn == 0
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4392.pdf

- Improved std::future<T> and Related API

```
// extensions to future<>
auto fut1 = async(launch::deferred, [] { return 1; });
auto fut2 = fut1.then([](auto x) { return 2; });
fut2.wait();
// composition of futures
auto meta_fut = when_all(fut1, fut2, fut3, ..., fut_n);
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4399.html

- Atomic Smart Pointers

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4399.html#atomic

- Variadic `lock_guard<>`

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4470.html

Parallelism 1/2

- Parallel Execution Policies

```
sort(seq, v.begin(), v.end()); // explicitly sequential sort
sort(par, v.begin(), v.end()); // allow parallel execution
sort(par_vec, v.begin(), v.end()); // allow vectorisation also
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4407.html#parallel.execpol

- Parallel Algorithms

```
std::vector<int> v;
for_each(par, std::begin(v), std::end(v), [&](int i) {
  /* ... code: keep data races, deadlocks, ... in mind */
});
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4407.html#parallel.alg

- Parallel Exceptions

```
try {
   transform(par, std::begin(v), std::end(v), std::begin(o));
} catch (exception_list&) {}
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4407.html#parallel.exceptions

Parallelism 1/2

adjacent_difference	adjacent_find	all_of	any_of
copy	copy_if	copy_n	count
count_if	equal	exclusive_scan	fill
fill_n	find	find_end	find_first_of
find_if	find_if_not	for_each	for_each_n
generate	generate_n	includes	inclusive_scan
inner_product	inplace_merge	is_heap	is_heap_until
is_partitioned	is_sorted	is_sorted_until	lexicographical_compare
max_element	merge	min_element	minmax_element
mismatch	move	none_of	nth_element
partial_sort	partial_sort_copy	partition	partition_copy
reduce	remove	remove_copy	remove_copy_if
remove_if	replace	replace_copy	replace_copy_if
replace_if	reverse	reverse_copy	rotate
rotate_copy	search	search_n	set_difference
set_intersection	set_symmetric_difference	set_union	sort
stable_partition	stable_sort	swap_ranges	transform
transform_exclusive_scan	transform_inclusive_scan	transform_reduce	uninitialized_copy
uninitialized_copy_n	uninitialized_fill	uninitialized_fill_n	unique
unique_copy			

Parallelism 1/2

- Parallel Execution Policies

```
sort(seq, v.begin(), v.end()); // explicitly sequential sort
sort(par, v.begin(), v.end()); // allow parallel execution
sort(par_vec, v.begin(), v.end()); // allow vectorisation also
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4407.html#parallel.execpol

- Parallel Algorithms

```
std::vector<int> v;
for_each(par, std::begin(v), std::end(v), [&](int i) {
  /* ... code: keep data races, deadlocks, ... in mind */
});
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4407.html#parallel.alg

- Parallel Exceptions

```
try {
   transform(par, std::begin(v), std::end(v), std::begin(o));
} catch (exception_list&) {}
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4407.html#parallel.exceptions

Parallelism 2/2

- Executors for Parallel Algorithms

```
// permit vector execution in the current thread only
sort(this_thread::vec, data.begin(), data.end());
// custom executors for fine-grained control
my_executor my_exec = ...;
std::sort(par.on(my_exec), data.begin(), data.end());
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4406.pdf

- Task Blocks *(pretty much PPL/TBB as part of STL)*

```
template <typename Func>
int traverse(node *n, Func&& compute) {
    int left = 0, right = 0;
    define_task_block([&](task_block& tb) {
        if (n->left)
            tb.run([&] { left = traverse(n->left, compute); });
        if (n->right)
            tb.run([&] { right = traverse(n->right, compute); });
    }); // synchronise with spawned tasks
    return compute(n) + left + right;
}
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4411.pdf

Core Language Additions 1/4

- Resumable Expressions (Stackless Co-Routines)

```
resumable void print_1_to(int n) {
    for(int i = 1; i < n; ++i) {
        std::cout << i << std::endl;
        break resumable;
    }
}
resumable auto r = print_1_to(5);
while (!r.ready()) {
    std::cout << "resuming... " << r.resume() << std::endl;
}
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4453.pdf

- `static_if` Resurrected

```
template <class T, class... Rest>
void f(T&& t, Rest&&... r) {
    f(t);
    static_if (sizeof...(r)) {
        f(r...); // No need for zero-param overload of f()
    }
}
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4461.html

Core Language Additions 2/4

- Template Parameter Deduction for Constructors

```
std::pair p(2, 4.5);  // omitting std::pair<int, double>
template <typename T> class Foo { Foo(T x) {} };
Foo f([&](int i) {}); // omitting Foo<???>
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4471.html

- `goto` in `constexpr` Functions

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4472.html

- Operator Dot

```
template <class X>
class Ref {
 public:
   X& operator.() { /* some code */ return *p; }

};
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4477.pdf

Core Language Additions 3/4

- Generalized Dynamic Assumptions

```
void bar(float *q, const float *p, int n, int m) {
    true(alignof(p) == 16 && alignof(q) == 16);
    true(m % 16 == 0);
    // This loop can safely be interleaved (modulo unrolled)
    // and/or vectorized by some factor no greater than 16,
    // based on the information provided
    for (int i = 0; i < n; ++i)
        q[i] = p[i] + p[i+m];
}
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4425.pdf

- Template Argument Type Deduction

```
template <typename T, T v> struct S;    // declaration
S<decltype(x), x> s;                    // instantiation (today)
S<x> s;                                 // desired instantiation
template <auto v> struct S;             // proposed declaration
template using typename T <T v> struct S; // - with explicit T
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4469.html

Core Language Additions 4/4

- Aggregate Init for Default Constructible Base Classes

```
struct Foo {};
struct Bar : Foo {
  int    a;
  double b;
};

Bar bar{1, 33.7};
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4404.html

- Inline Variables

```
struct WithStaticDataMember {
   // no out-of-line definition is required
   static inline constexpr const char *kFoo = "foo bar";
};
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4424.pdf

Modules

"It should be a matter of mechanical replacement of header files with corresponding module import declarations and nothing else."

- Module System for C++

```
import std.vector; // #include <vector>
import std.string; // #include <string>

int main() {
    using namespace std;
    vector<string> v = {"Socrates", "Plato", "Kant", "Bacon"};
}
```

```
module M;
export {
    int f(int);
    double g(double, int);
}
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4465.pdf

Reflection and Testing

- Static Reflection *(in collaboration with Axel - CERN)*

```
struct A {
   typedef mirrored(this::class) meta_A; // MetaClass
   int a;
};
typedef mirrored(A::a) meta_A_a;      // MetaClassMember
typedef mirrored(std::sin) meta_sin; // MetaOverloadedFunction
assert (strcmp(meta::full_name<meta_A_a>(), "A::a") == 0);
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4451.pdf

- Source Code Information Capture

```
std::source_context sc(); // capture source code information
std::cerr << "ERROR: at " << sc.line_number()
          << " in file "  << sc.file_name()   << std::endl;
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4417.pdf

- Defining Test Code Inline

```
int foo() { return 0; }
struct FooFixture test { void testFoo() test; };
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4420.pdf

Contracts

- Language Support for Contract Programming

```
double square_root(double x) expects {x >= 0}
                             ensures {square_root >= 0.0};
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2014/n4293.pdf

- Simple Contracts

```
T& operator[](size_t i)          [[expects: i < size()]];
ArrayView(const vector<T>& v) [[ensures: data()== v.data()]];
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4415.pdf

- Language Support for Contract Assertions

```
bool binary_search(...) {
  contract_assert (table != nullptr);
  contract_assert_max (std::is_sorted(table, table + size));
  while (size != 0) {
    contract_assert (candidate <= table[size - 1));
    /* ... implementation goes here */
  }
}
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4378.pdf

Thanks!

Questions?
(more) Discussion?


CERN