An Oddysey in C++

A Guided Tour of Modern C++

Per Karlström
per@karlstrom.se

Upplysningen 2010-03-02

From C to C++

Bjarne Stroustrup
Father of C++

- **1979**: Development starts
- **1998**: ISO/IEC 14882:1998
- **2003**: ISO/IEC 14882:2003
- **2005**: TR1

From C to C++

Bjarne Stroustrup
Father of C++

- **1979**: Development starts
- **1998**: ISO/IEC 14882:1998
- **2003**: ISO/IEC 14882:2003
- **2005**: TR1

From C to C++

Bjarne Stroustrup
Father of C++

- **1979**: Development starts
- **1998**: ISO/IEC 14882:1998
- **2003**: ISO/IEC 14882:2003
- **2005**: TR1

From C to C++

Bjarne Stroustrup
Father of C++

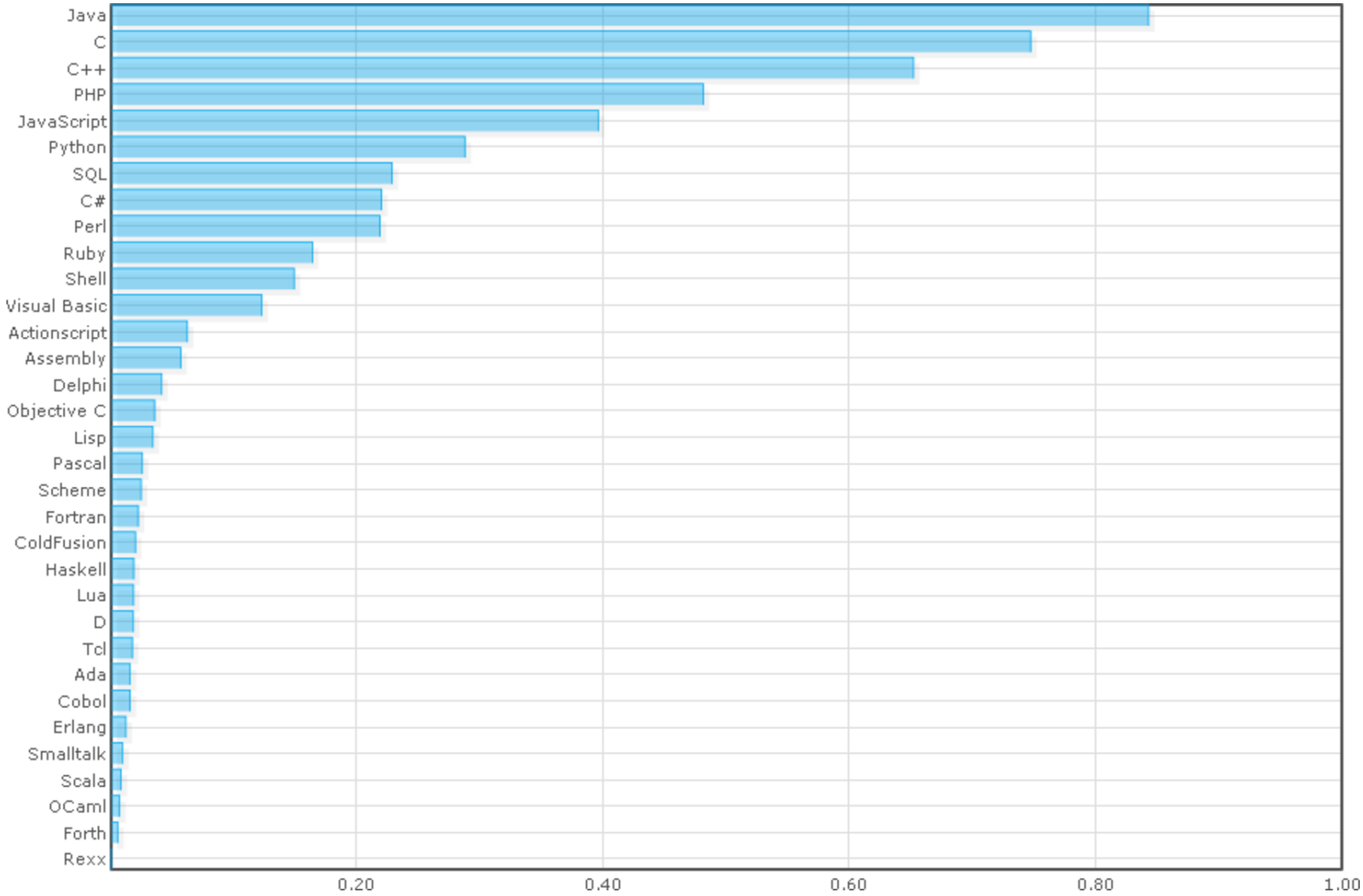
- **1979**: Development starts
- **1998**: ISO/IEC 14882:1998
- **2003**: ISO/IEC 14882:2003
- **2005**: TR1

From C to C++

Bjarne Stroustrup
Father of C++

- **1979**: Development starts
- **1998**: ISO/IEC 14882:1998
- **2003**: ISO/IEC 14882:2003
- **2005**: TR1

Current State of C++



http://langpop.com/

Hello Template World

template<typename T>...

Complex Numbers

```
class Complex{
  float re,im;
public:
  ...
  const float& re() const;
  const float& im() const;
};
```

Complex Template Numbers

```
template<typename T>
class Complex{
  T re,im;
public:
  ...
  const T& re() const;
  const T& im() const;
};
```

Advanced Template Techniques

- Template specialization
- Template template arguments
- SFINAE

Advanced Template Techniques

- Template specialization
- Template template arguments
- SFINAE

Advanced Template Techniques

- Template specialization
- Template template arguments
- SFINAE

Standard Template Library (STL)

- Containers
- Iterators
- Algorithms
- Functors

Standard Template Library (STL)

- Containers
- Iterators
- Algorithms
- Functors

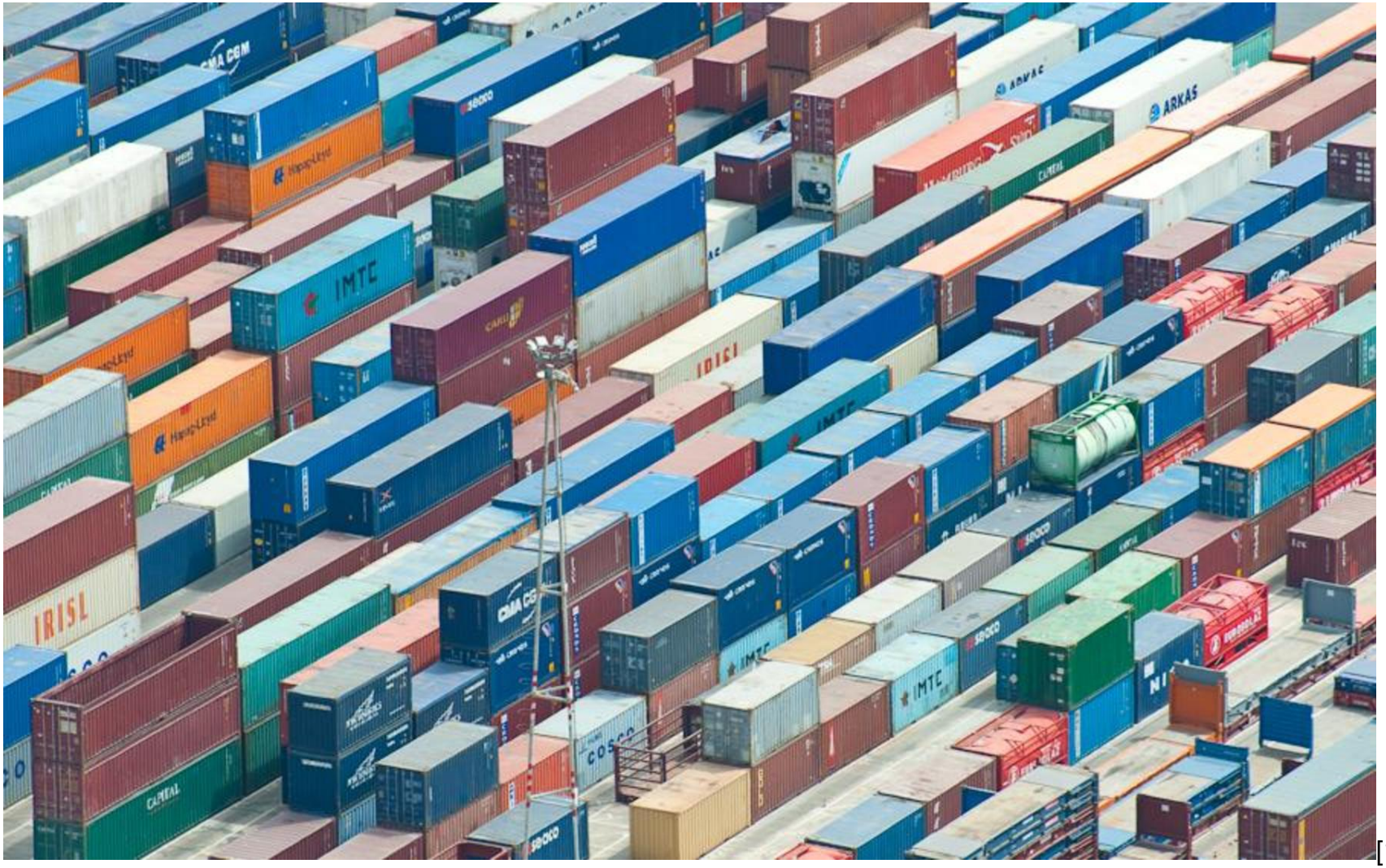
Standard Template Library (STL)

- Containers
- Iterators
- Algorithms
- Functors

Standard Template Library (STL)

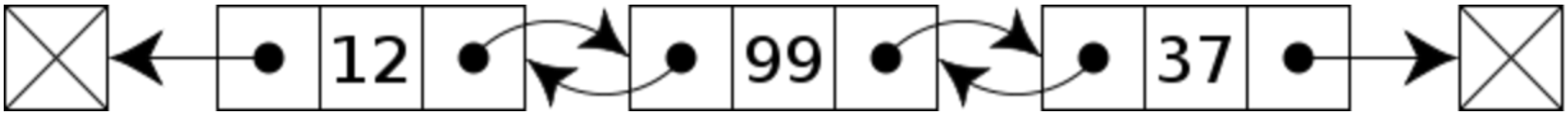
- Containers
- Iterators
- Algorithms
- Functors

Containers

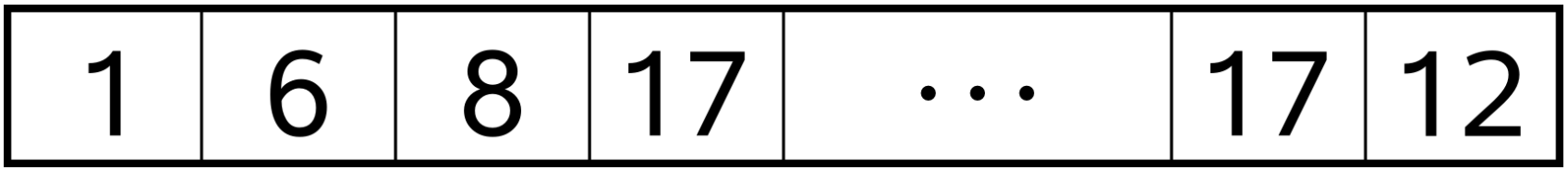


[1]

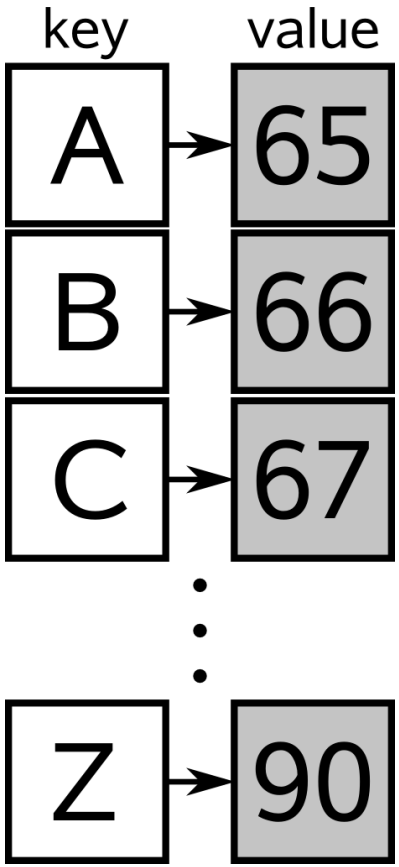
List



Vector



Map



String

```
#include <string>
...
std::string better_than_charp("hello string");
```

```
typedef std::basic_string<char> string;
```

String

```
#include <string>
...
std::string better_than_charp("hello string");
```

```
typedef std::basic_string<char> string;
```

Other Containers

- Deque
- Stack
- Queue
- Multimap
- Set, Multiset

Iterators

Different types of Iterators

- Trivial
- Input
- Output
- Forward
- Bidirectional
- Random Access

Iterators

Different types of Iterators

- Trivial
- Input
- Output
- Forward
- Bidirectional
- Random Access

Iterators

Different types of Iterators

- Trivial
- Input
- Output
- Forward
- Bidirectional
- Random Access

Iterators

Different types of Iterators

- Trivial
- Input
- Output
- Forward
- Bidirectional
- Random Access

Iterators

Different types of Iterators

- Trivial
- Input
- Output
- Forward
- Bidirectional
- Random Access

Iterators

Different types of Iterators

- Trivial
- Input
- Output
- Forward
- Bidirectional
- Random Access

Algorithms for STL Containers

- Non mutating
  - For each
  - Find
  - ...
- Mutating
  - Copy
  - Transform
  - ...
- Sorting
  - Sort
  - Binary search
  - Heap operations
  - ...
- Numeric

Algorithms for STL Containers

- Non mutating
  - For each
  - Find
  - ...
- Mutating
  - Copy
  - Transform
  - ...
- Sorting
  - Sort
  - Binary search
  - Heap operations
  - ...
- Numeric

Algorithms for STL Containers

- Non mutating
  - For each
  - Find
  - ...
- Mutating
  - Copy
  - Transform
  - ...
- Sorting
  - Sort
  - Binary search
  - Heap operations
  - ...
- Numeric

Algorithms for STL Containers

- Non mutating
  - For each
  - Find
  - ...
- Mutating
  - Copy
  - Transform
  - ...
- Sorting
  - Sort
  - Binary search
  - Heap operations
  - ...
- Numeric

For each

Example

STL for each...

For each

Example

STL for each...

Copy

Example

STL copy...

Copy

Example

STL copy...

Transform

Example

STL transform...

Transform

Example

STL transform...

Sort

Example

STL sort...

Sort

Example

STL sort...

Boost

"...one of the most highly regarded and expertly designed C++ library projects in the world."
— —Herb Sutter and Andrei Alexandrescu, C++ Coding Standards

- Initial proposal 1998 by Beman G. Daves
- 16 804 806 LOC
- 5293 Person Years
- $291 133 309

Boost

"...one of the most highly regarded and expertly designed C++ library projects in the world."
— —Herb Sutter and Andrei Alexandrescu, C++ Coding Standards

- Initial proposal 1998 by Beman G. Daves
- 16 804 806 LOC
- 5293 Person Years
- $291 133 309

Boost

"...one of the most highly regarded and expertly designed C++ library projects in the world."
— —Herb Sutter and Andrei Alexandrescu, C++ Coding Standards

- Initial proposal 1998 by Beman G. Daves
- 16 804 806 LOC
- 5293 Person Years
- $291 133 309

Boost

"...one of the most highly regarded and expertly designed C++ library projects in the world."
— —Herb Sutter and Andrei Alexandrescu, C++ Coding Standards

- Initial proposal 1998 by Beman G. Daves
- 16 804 806 LOC
- 5293 Person Years
- $291 133 309

Boost

"...one of the most highly regarded and expertly designed C++ library projects in the world."
— —Herb Sutter and Andrei Alexandrescu, C++ Coding Standards

- Initial proposal 1998 by Beman G. Daves
- 16 804 806 LOC
- 5293 Person Years
- $291 133 309

BOOST_FOREACH

Example

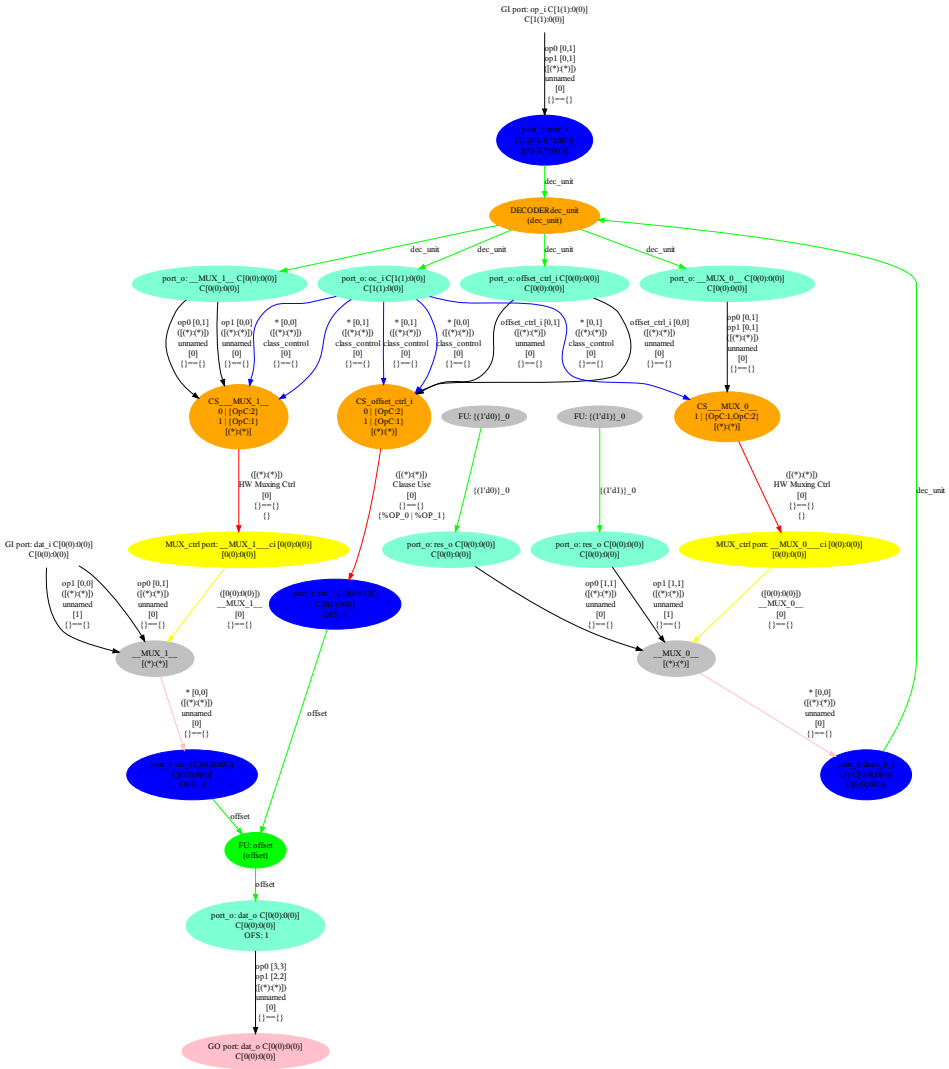
Boost for each...

BOOST_FOREACH

Example

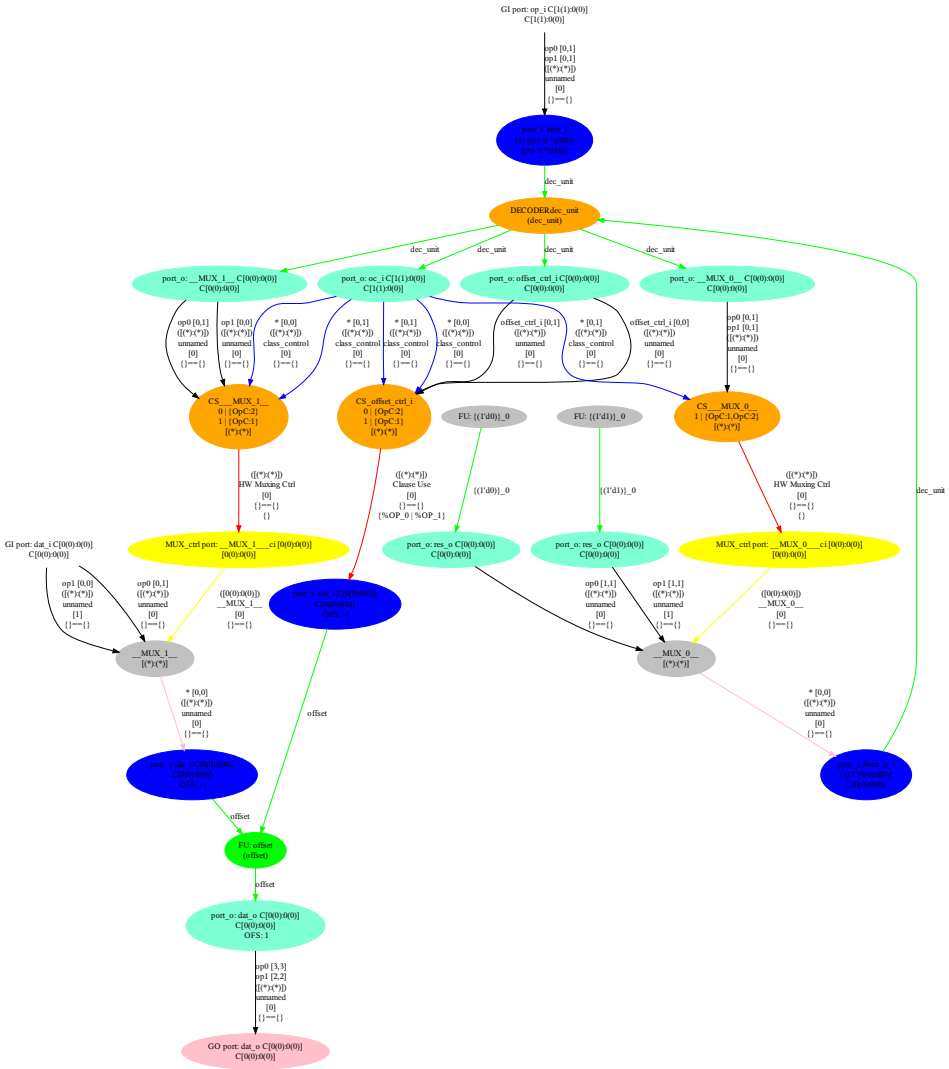
Boost for each...

Boost Graph Library



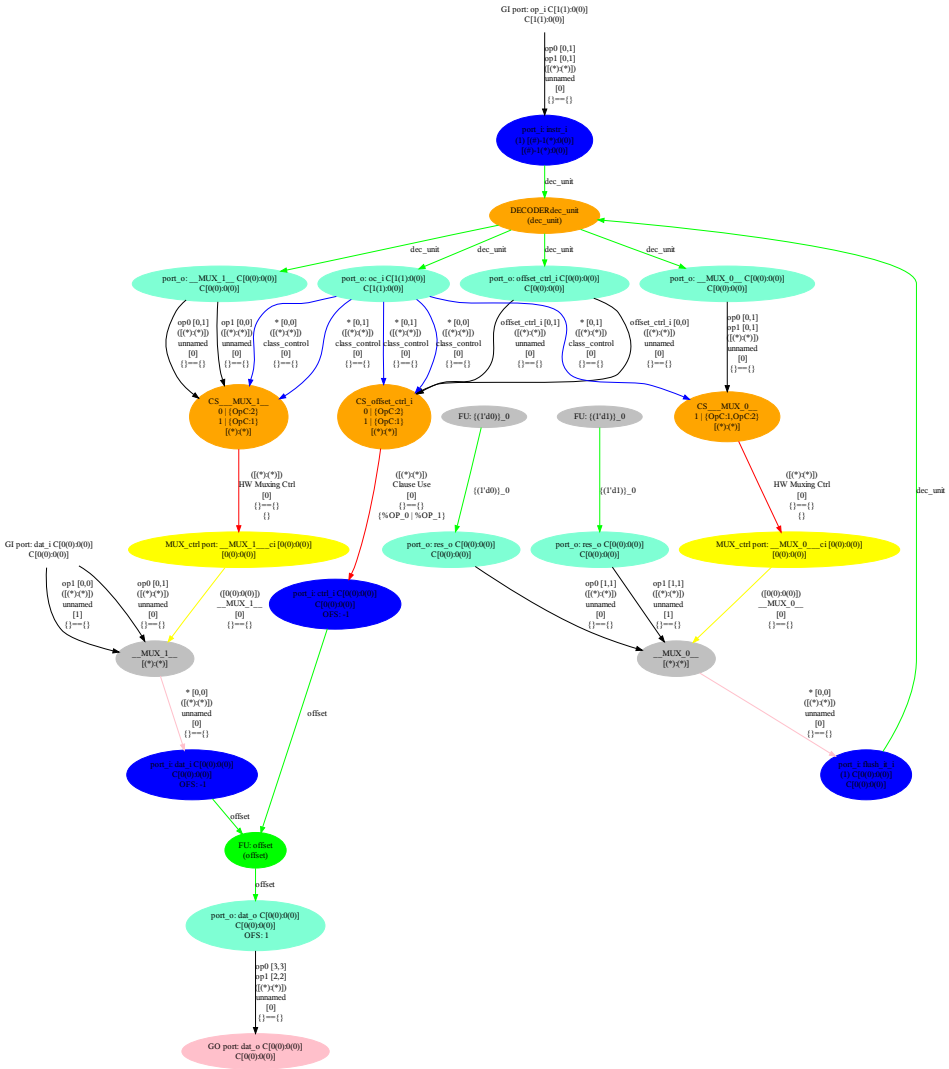
- General Graph Library
- Graph I/O with Graphwiz
- Graph Algorithms

Boost Graph Library



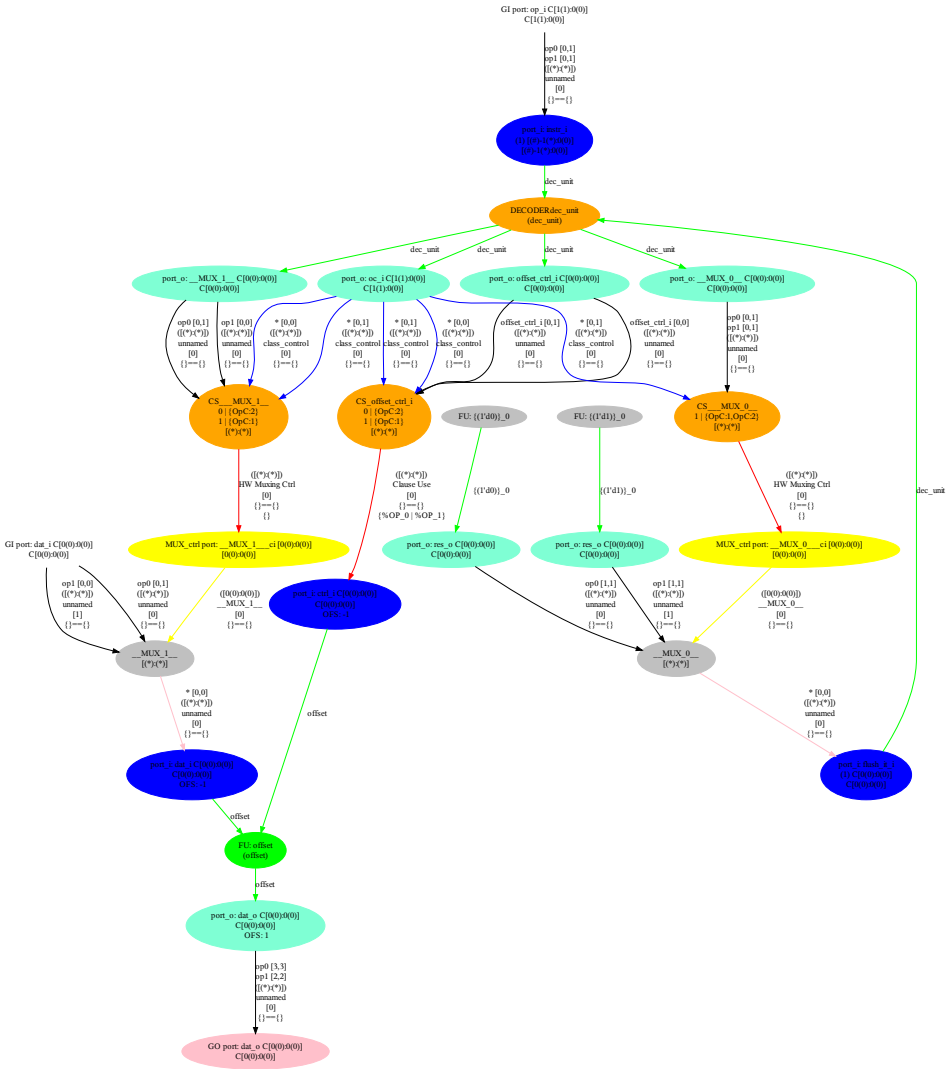
- General Graph Library
- Graph I/O with Graphwiz
- Graph Algorithms

Boost Graph Library



- General Graph Library
- Graph I/O with Graphwiz
- Graph Algorithms

Boost Graph Library



- General Graph Library
- Graph I/O with Graphwiz
- Graph Algorithms

Regex

Example

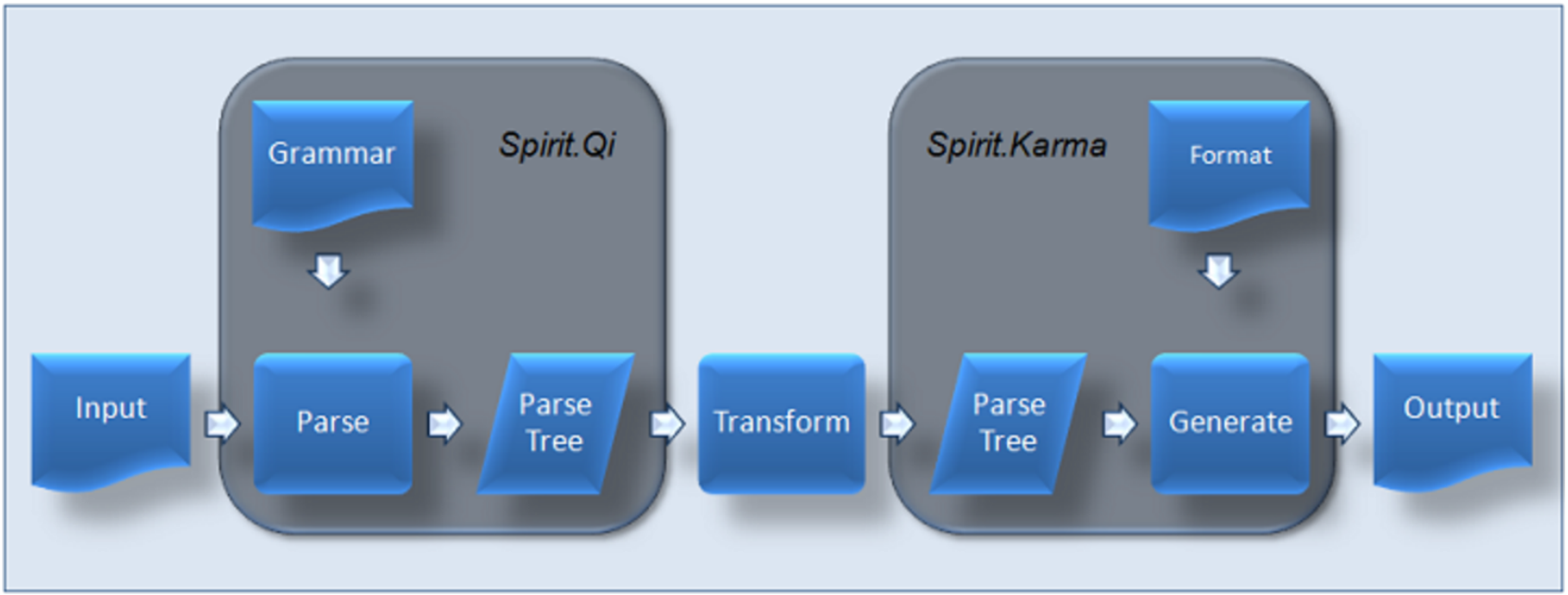
Credit card number parsing...

Regex

Example

Credit card number parsing...

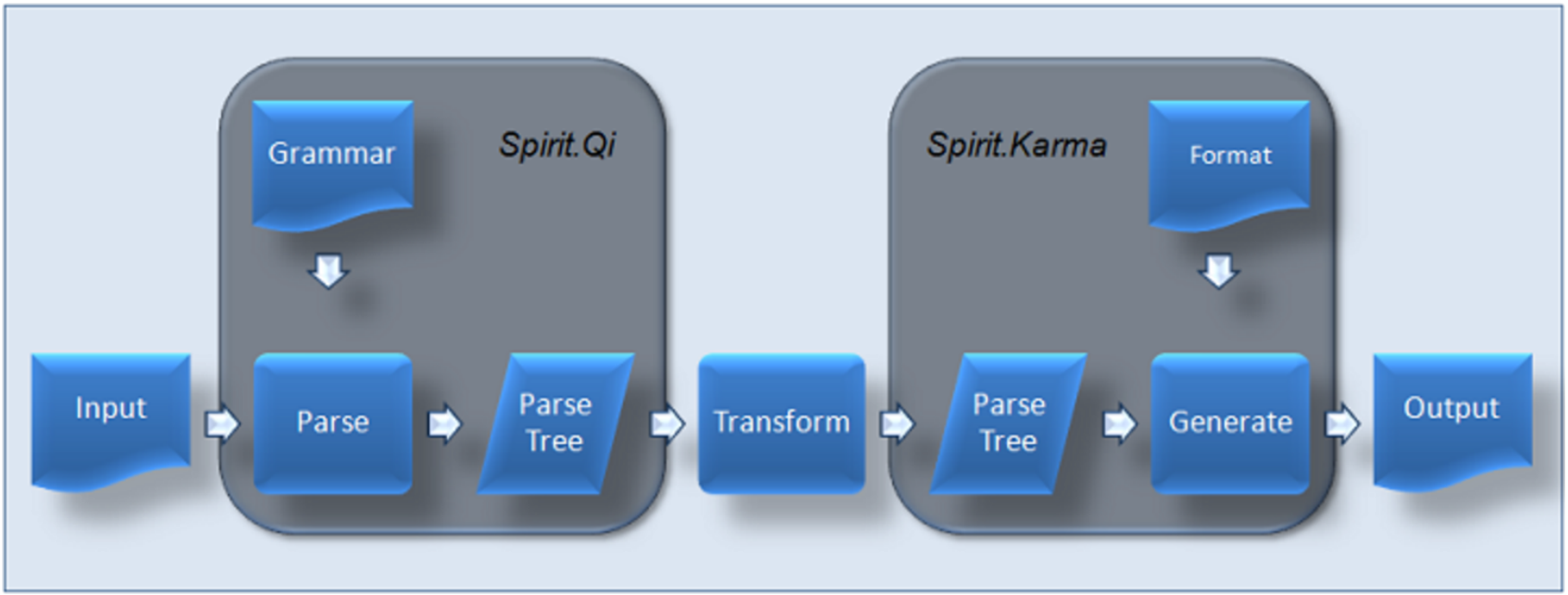
Spirit



Example

Roman number parser...

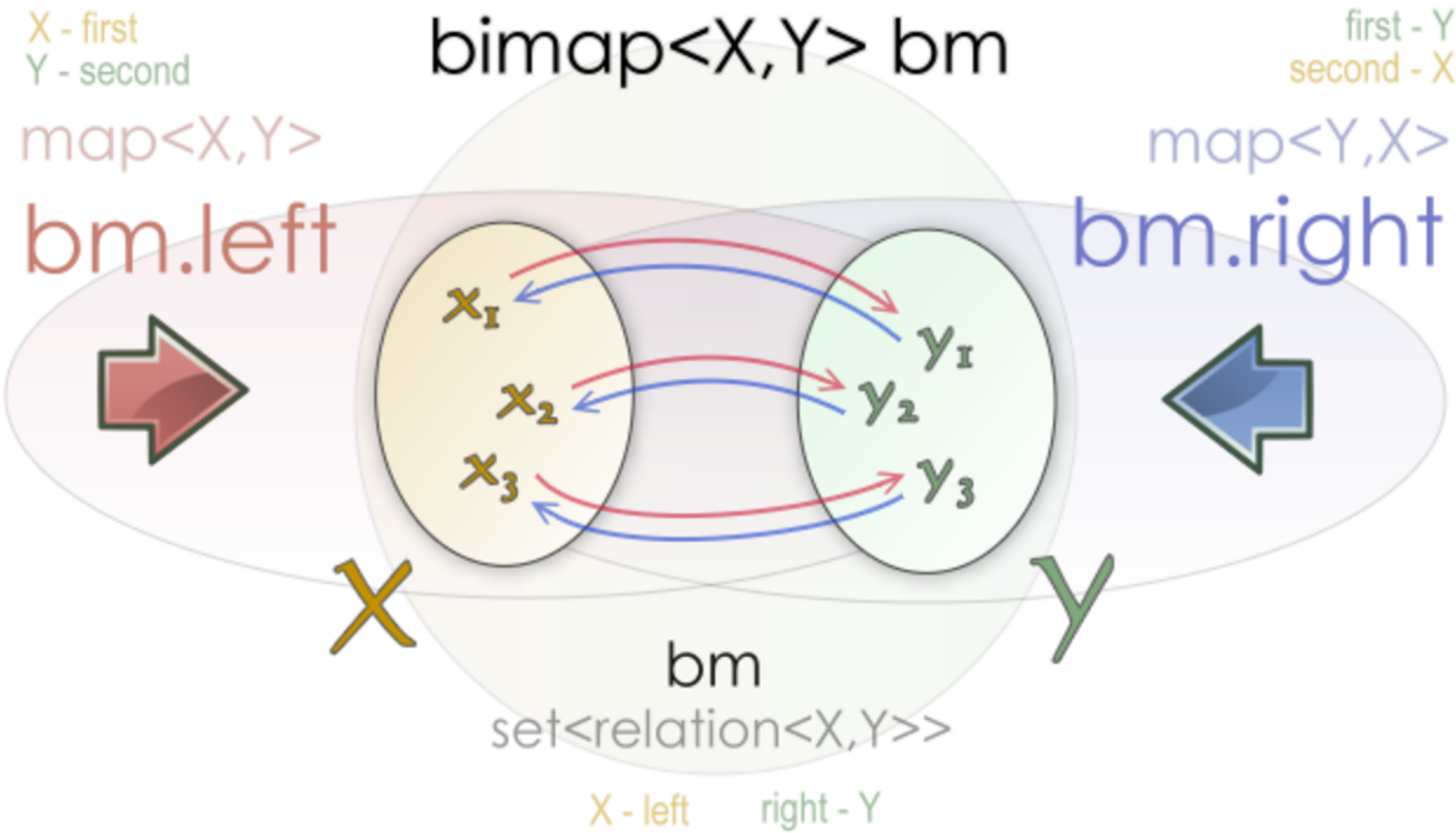
Spirit



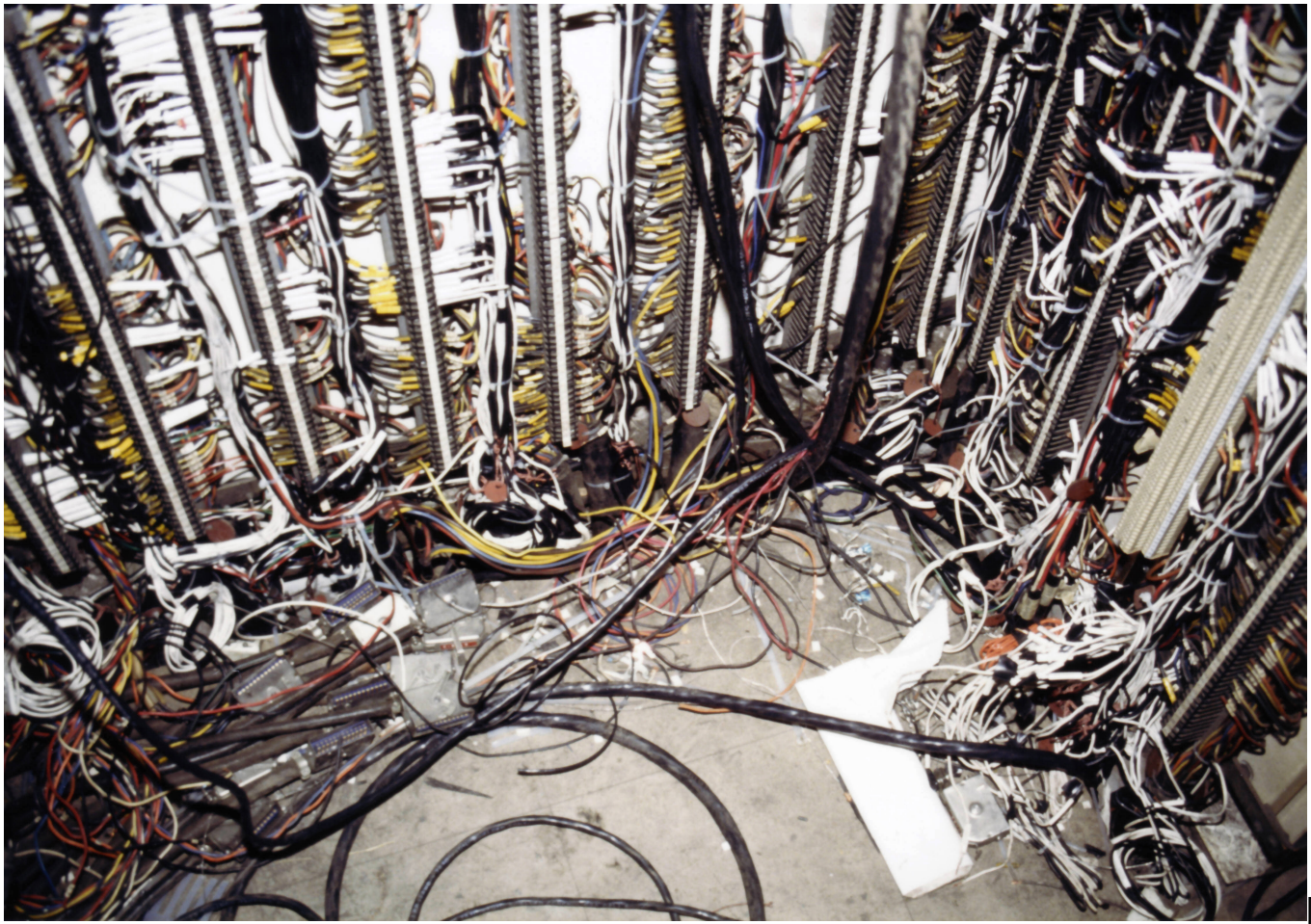
Example

Roman number parser...

Bimap


X - first
Y - second
map<X,Y>
bm.left
bimap<X,Y> bm
first - Y
second - X
map<Y,X>
bm.right
x_1
x_2
x_3
y_1
y_2
y_3
X
Y
bm
set<relation<X,Y>>
X - left
right - Y

ASIO



[2]

Format

Printf like format specification for `std::cout`

Example

Some formatted output...

Format

Printf like format specification for `std::cout`

Example

Some formatted output...

Interprocess

Cross platform interprocess communication.

- Shared memory
- Memory-mapped files
- Semaphores, mutexes,
- Named synchronization objects
- File locking
- Relative pointers
- Message queues

λ-functions in C++

```
int foo(int)
...
list<int> v(10);
for_each(v.begin(), v.end(), _1 = 17);
vector<int*> vp(10);
transform(v.begin(),v.end(),vp.begin(),&_1);
for_each(v.begin(),v.end(), _1 = bind(foo,_1));
sort(vp.begin(),vp.end(),*_1 < *_2);
for_each(vp.begin(),vp.end(), cout << *_1 << '\n');
```

Optional

Problem

```
int read_int(std::istream&);
```

Traditional Solutions

```
bool read_int(std::istream&, int&);
int read_int(std::istream&) throw(some error);
```

Solution with boost::optional

```
boost::optional<int> read_int(std::istream&);
```

Optional

Problem

```
int read_int(std::istream&);
```

Traditional Solutions

```
bool read_int(std::istream&, int&);
int read_int(std::istream&) throw(some error);
```

Solution with boost::optional

```
boost::optional<int> read_int(std::istream&);
```

Optional

Problem

```
int read_int(std::istream&);
```

Traditional Solutions

```
bool read_int(std::istream&, int&);
int read_int(std::istream&) throw(some error);
```

Solution with boost::optional

```
boost::optional<int> read_int(std::istream&);
```

Program Options

Example

Program options...

Program Options

Example

Program options...

Serialization

- Portable
- Versioning
- Proper pointer store/restore
- Serialization of STL containers

Example

Serialization example...

Serialization

- Portable
- Versioning
- Proper pointer store/restore
- Serialization of STL containers

Example

Serialization example...

Smart Pointres

Traditional Solution

```
int* int_pointer = new int(17);
....
// Somewhere else in the code
delete int_pointer
```

Problems

- Forget `delete`
- Miss `delete` due to exception
- Memory management code

Solution with boost::shared_ptr

```
boost::shared_ptr<int> int_pointer(new int(17));
```

Smart Pointres

Traditional Solution

```
int* int_pointer = new int(17);
....
// Somewhere else in the code
delete int_pointer
```

Problems

- Forget `delete`
- Miss `delete` due to exception
- Memory management code

Solution with boost::shared_ptr

```
boost::shared_ptr<int> int_pointer(new int(17));
```

Smart Pointres

Traditional Solution

```
int* int_pointer = new int(17);
....
// Somewhere else in the code
delete int_pointer
```

Problems

- Forget `delete`
- Miss `delete` due to exception
- Memory management code

Solution with boost::shared_ptr

```
boost::shared_ptr<int> int_pointer(new int(17));
```

Smart Pointres

Traditional Solution

```
int* int_pointer = new int(17);
....
// Somewhere else in the code
delete int_pointer
```

Problems

- Forget `delete`
- Miss `delete` due to exception
- Memory management code

Solution with boost::shared_ptr

```
boost::shared_ptr<int> int_pointer(new int(17));
```

Smart Pointres

Traditional Solution

```
int* int_pointer = new int(17);
....
// Somewhere else in the code
delete int_pointer
```

Problems

- Forget `delete`
- Miss `delete` due to exception
- Memory management code

Solution with boost::shared_ptr

```
boost::shared_ptr<int> int_pointer(new int(17));
```

Other Smart Pointers

- scoped_ptr
- scoped_array
- shared_array
- weak_ptr
- intrusive_ptr

Other Smart Pointers

- scoped_ptr
- scoped_array
- shared_array
- weak_ptr
- intrusive_ptr

Other Smart Pointers

- scoped_ptr
- scoped_array
- shared_array
- weak_ptr
- intrusive_ptr

Other Smart Pointers

- scoped_ptr
- scoped_array
- shared_array
- weak_ptr
- intrusive_ptr

Other Smart Pointers

- scoped_ptr
- scoped_array
- shared_array
- weak_ptr
- intrusive_ptr

And Much Much More...

- Accumulators
- CRC
- Date and time
- Lexical cast
- Math
- Signals
- State chart
- Units
- Template meta programming
- Random
- Generic Image Library
- In_place_factory
- Message Passing Interface
- Even more at `www.boost.org`

C++0x - The Next C++ Standard

- Expected to be done by 2011
- Almost 100% backward compatible
- Support is growing

C++0x - The Next C++ Standard

- Expected to be done by 2011
- Almost 100% backward compatible
- Support is growing

C++0x - The Next C++ Standard

- Expected to be done by 2011
- Almost 100% backward compatible
- Support is growing

Rvalue Reference

- Move semantics
- Perfect Forwarding

Example

Vector move...

Rvalue Reference

- Move semantics
- Perfect Forwarding

Example

Vector move...

Rvalue Reference

- Move semantics
- Perfect Forwarding

Example

Vector move...

Generalized Constants

Illegal in ANSI C++

```
int GetFive() {return 5};
...
int arr[GetFive()+7];
```

Legal in C++0x

```
constexpr int GetFive() {return 5};
...
int arr[GetFive()+7];
```

Generalized Constants

Illegal in ANSI C++

```
int GetFive() {return 5};
...
int arr[GetFive()+7];
```

Legal in C++0x

```
constexpr int GetFive() {return 5};
...
int arr[GetFive()+7];
```

Initializer Lists

```
class SequenceClass{
public:
  SequenceClass(std::initializer_list<int> list);
};
...
SequenceClass someVar = {1, 4, 5, 6};

void FunctionName(std::initializer_list<float> list);
FunctionName({1.0f, -3.45f, -0.4f});

std::vector<std::string> v = { "xy", "plu", "abra" };
std::vector<std::string> v{ "xy", "plu", "abra" };
```

Uniform Initialization

```
struct BasicStruct{
 int x;
 double y;
};
struct AltStruct{
  AltStruct(int x, double y) : x_{x}, y_{y} {}
private:
  int x_;
  double y_;
};
BasicStruct var1{5, 3.2};
AltStruct var2{2, 4.3};
```

Automatic Types

```
auto someType = boost::bind(&someFunction, _2,
                                    _1, someObject);
auto otherVariable = 5;

int someInt;
decltype(someInt) otherIntegerVariable = 5;
```

Range Based For Loops

Works for

- arrays
- initializer lists
- Containers with begin() ... end()

```
int my_array[5] = {1, 2, 3, 4, 5};
for(int& x : my_array)
{
  x *= 2;
}
```

λ-functions in C++

```
int foo(int)
...
list<int> v(10);
for_each(v.begin(), v.end(), _1 = 17);
vector<int*> vp(10);
transform(v.begin(),v.end(),vp.begin(),&_1);
for_each(v.begin(),v.end(), _1 = bind(foo,_1));
sort(vp.begin(),vp.end(),*_1 < *_2);
for_each(vp.begin(),vp.end(), cout << *_1 << '\n');
```

Lambda Functions

```
[](int x, int y) { return x + y; }
[](int x, int y) -> int { int z = x + y; return z + x; }
std::vector<int> someList;
int total = 0;
std::for_each(someList.begin(),
              someList.end(), [&total](int x) {
  total += x;
});
std::cout << total;
```

Alternative Function Syntax

Problem

```
template< typename LHS, typename RHS>
Ret AddingFunc(const LHS &lhs, const RHS &rhs)
{return lhs + rhs;}
```

Solution

```
template< typename LHS, typename RHS>
auto AddingFunc(const LHS &lhs, const RHS &rhs)
-> decltype(lhs+rhs)
{return lhs + rhs;}
```

Alternative Function Syntax

Problem

```
template< typename LHS, typename RHS>
Ret AddingFunc(const LHS &lhs, const RHS &rhs)
{return lhs + rhs;}
```

Solution

```
template< typename LHS, typename RHS>
auto AddingFunc(const LHS &lhs, const RHS &rhs)
-> decltype(lhs+rhs)
{return lhs + rhs;}
```

Varadic Templates

```
template<typename T, typename... Args>
void printf(const char* s, T value, Args... args)
{
  while (*s)
  {
    if (*s == '%' && *(++s) != '%')
    {
      std::cout << value;
      printf(s, args...);
      return;
    }
    std::cout << *s++;
  }
  throw std::logic_error
        ("extra arguments provided to printf");
}
```

User Defined Literals

```
OutputType operator "" _Suffix(unsigned long long);
OutputType operator "" _Suffix(long double);

OutputType someVariable    = 1234_Suffix;
OutputType anotherVariable = 3.1416_Suffix;
```

Default and Delete Member Functions

```
struct NonCopyableAndNewable
{
  NonCopyable & operator=(const NonCopyable&) = delete;
  NonCopyable(const NonCopyable&) = delete;
  NonCopyable() = default;
  void *operator new(std::size_t) = delete;

  void f(int i);
  template<class T> void f(T) = delete;
};
```

Compile Time Assertion

Example

Static assert example...

Compile Time Assertion

Example

Static assert example...

Other Changes

- Relaxed POD requirements
- extern template
- `nullptr`
- Strongly typed enumerations
- Explicit conversion operators
- Unrestricted unions
- Unicode strings
- Multithreading

More Reading

Bjarne Stroustrup

The C++ Programming Language

Nicolai M. Josuttis

The C++ Standard Library: A Tutorial and Reference

David Vandevoorde, Nicolai M. Josuttis

C++ Templates: The Complete Guide

Boost

`http://www.boost.org`

STL

`http://www.sgi.com/tech/stl/`

C++0x

`http://en.wikipedia.org/wiki/C%2B%2B0x`

Questions

Questions

Attribution

[1] [1]Christian Coo
http://commons.wikimedia.org/wiki/File:Puertobarcelona2.jpg

[2] [2]Achim Hering
http://commons.wikimedia.org/wiki/File:Cable_sald.jpg