Outline

1. Overview
2. History
   - C with Classes
   - Early C++
   - ARM C++
   - Standard Template Library
   - ISO C++: C++98
3. Future
   - C++0x
   - Afterwards
4. References
Overview

Basics of C++

- Designed (mostly) by Bjarne Stroustrup.
  - Begun as PhD work in Cambridge University, England.
- From the *C++ Programming Language* book: *C++ is a general purpose programming language designed to make programming more enjoyable for the serious programmer.*
- Goal: Simula’s facilities for program organization with C’s efficiency and flexibility for systems programming [4].
- Not (merely) an object-oriented language, but a multi-paradigm language.
- Static types and compile-time type checking.
- Separation: OS is kept separate from the language.
- Design now controlled by ISO C++ standards committee.
Origins

Stroustrup was working with Simula on his Ph.D. thesis in the Computing Laboratory of Cambridge University, England.

Languages at the time

- Simula was a good language, but the implementation was not efficient.
- BCPL was efficient, but too low-level for large-scale applications.
Usage

How many programmers use C++?

<table>
<thead>
<tr>
<th>Year</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1980</td>
<td>16</td>
</tr>
<tr>
<td>1985</td>
<td>500</td>
</tr>
<tr>
<td>1987</td>
<td>4000</td>
</tr>
<tr>
<td>1988</td>
<td>15000</td>
</tr>
<tr>
<td>1990</td>
<td>150000</td>
</tr>
<tr>
<td>1991</td>
<td>400000</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>3270000</td>
</tr>
</tbody>
</table>
From a survey by Tiobe Softwares in Jan 2009 [2]:
Usage

Where is C++ used? [5]

- Adobe Acrobat, Photoshop, Illustrator, …
- Amazon
- Apple iPod, finder, …
- eBay
- Games (Doom 3, StarCraft, Halo, …)
- Google search engines, Google Earth, …
- Maya
- Microsoft Windows XP, Office, IE, Visual Studio, .NET, …
- Mozilla Firefox
- Sun compilers, OpenOffice, HotSpot Java Virtual Machine
- Symbian OS

- C is *flexible, efficient, available, and portable*.

Why not a completely separate language?

- Provides easy transition from and interoperability with C.
- Possible to implement as a C front-end (e.g., Cfront \(^a\)).
- Available *now, or next month*, rather than *in two years*.
- Use standard linker.
- Interoperability with other languages.
- Close to the machine.

\(^a\)In Cfront, C compiler is used only as a code generator.
History of C++

What is the history of the C++ language?
C with Classes

- Implemented as a C preprocessor in 1980.
- Simple classes and class hierarchies for organizing data.
  - Derived and base classes instead of sub- and super-
- Classes could have member functions, constructors, destructors, etc.
- Not claimed to support object-oriented programming.
- No virtual functions, templates, or any other advanced features of C++.
- Also had call and return functions.
  - Class functions called before and after calls to any other class functions.
- Inline functions, default arguments and assignment operator overloading added in 1981.
In 1982, Stroustrup realized that C with Classes was a medium success, and would remain so.

- C with Classes did help organize a large class of problems better than C,
- but it was not attractive enough to continue developing and supporting.

The new extended language was first named C84, but this was deemed too institutional.

The name C++ was suggested by Rick Mascitti in 1983.
Some first features of C++

Additions

- Virtual functions
  - From Simula, to prevent `switch(type)` style constructs
- Function name and operator overloading
  - Experience of operator overloading from Algol68
- References
  - Provided to support operator overloading, experiences from Algol68
- Constants

The notion of *call and return functions* was removed from the language because it was not used.
The stream I/O library

- Created as a type-safe, extensible and efficient alternative to the printf function.

Example

In classic C:

```c
/* Primitive types */
fprintf(stderr, "x = %.2f\n", x);

/* Custom types */
fprintf(stderr, "x = ");
put_complex(stderr, x);
fprintf(stderr, "\n");
```

In C++:

```cpp
/* All types */
cerr << "x = " << fixed << setprecision(2) << x << "\n";
```

Java (old style):

```java
/* Primitive types */
NumberFormat n = new DecimalFormat("#.##");
System.out.println("x = " + n.format(x) + "\n");
```
The “whatis?” paper

- Listed missing features of C++:
  - Parametrized types
  - Exceptions
  - Multiple inheritance

Example of planned exceptions [3]

```cpp
class vector {
    void f() { if (...) raise vector_range; }
    except vector_range {
        error("global: vector range error;");
        exit(99); }
};
void f() {
    vector v; try { v.f(); }
    except { vector::vector_range:
        error("vector range error"); }
}
```
C++ 2.0

- Second major release of C++ in 1987.
- Some features:
  - Multiple inheritance
  - Abstract classes
  - Static and const member functions
- Featured a space optimization in comparison to the previous release, which had separate virtual function tables for each compilation unit.

The reference manual covers entire C++ (at the time), including planned features:
- Templates
- Exception handling

These were planned from the start, but had been postponed.

Includes annotations that explain why some features are designed as they are and how they might be implemented.

First “standard” for the C++ language.
Standard Template Library

- A container library supporting generic programming.
- Basic idea was to create the most general and most efficient code.
- Stroustrup found out about Alex Stepanov’s STL library in 1993.
  - Stepanov had worked with the ideas for over a decade before this.
- STL code may seem quite complicated (e.g., cryptic error messages).
  - It took weeks for Stroustrup to get comfortable with STL.
Container checklist

Stroustrup has developed a checklist of important properties for containers [5]:

- Individual containers are simple and efficient.
- Simple operations do not require function calls.
- Statically type safe and homogenous.
- Containers are non-intrusive.
- Containers can contain primitive types.
- ... Common services can be provided in a single place (e.g., base class).

STL fulfilled every point except the last one.
Simple usage

```cpp
vector<int>::iterator p = find(v.begin(), v.end(), 7);

/* This can be implemented with a template: */
while (first != last && *first != val)
    ++first;
```

The crucial point in this example is that this same template will work both with iterator types that are classes and with int pointers, for example.
### Lambda emulation

```cpp
list<int> lst;
Lambda x;
list<int>::iterator p = find_if(lst.begin(), lst.end(), x < 7);

/* This can be implemented with a template: */
while (first != last && !pred(*first))
    ++first;
```
Lambda emulation

```cpp
list<int> lst;
Lambda x;
list<int>::iterator p = find_if(lst.begin(), lst.end(), x < 7);

/* This can be implemented with a template: */
while (first != last && !pred(*first))
   ++first;

template<class T>
class Lambda {
   LessThan<T> operator<(const T & v) {
      return LessThan<T>(v);
   }
};
```
Lambda emulation

```cpp
list<int> lst;
Lambda x;
list<int>::iterator p = find_if(lst.begin(), lst.end(), x < 7);

/* This can be implemented with a template: */
while (first != last && !pred(*first))
  ++first;

template<class T>
class Lambda {
  LessThan<T> operator<(const T & v) {
    return LessThan<T>(v);
  }
};

class Lambda {};
template<class T>
LessThan<T> operator<(Lambda, const T & v) {
  return LessThan<T>(v);
}
```

The C++ standard [5, 6]

- Some new features:
  - Namespaces
  - Run-time type identification (RTTI)
  - Boolean type
- Includes the standard library, which includes the STL.

What was left out

- Concurrency (no clear idea what would be best approach, libraries can be used)
- More useful libraries (not enough resources, STL required much work)
- GUI (see previous)
C++ Standard Library

- C standard library
- The STL
- iostreams
- locales
- string
- bitset
- complex
- valarray
- auto_ptr
Updates to the standard

- Corrected version of the standard.
- Contains just minor corrections.

- *Technical Report on C++ Performance*
- Embedded C++, or EC++ for embedded systems programming.
- Subset of C++ (almost).

2005: The Library TR (TR1) (ISO/IEC PDTR 19768)
- *Technical Report on C++ Standard Library Extensions*
- Extensions to the standard libraries
- Tuple types, regexp, smart pointers, random, hash tables...
Future of C++

Future

What is the future of the C++ language?
Next C++ standard: C++0x [5, 7]

C++0x

- Will come out earliest at the end of 2011 (C++0B?)
- Some planned features:
  - Move semantics (`vector<int> a = f();`)
  - Generalized constant expressions (`constexpr`)
  - Initializer lists (`Sequence s = { 1, 4, 5, 6 };`)
  - Type inference (`auto, decltype`)
  - Range-based for-loop (`for(int &x : my_array)`)  
  - Lambda functions and expressions
    - `[](int x, int y) { return x + y; }`
  - Construction improvement (calls to peer constructors)
  - Null pointer constant (`nullptr`)
  - Variadic templates
    - `template<typename... Values> class tuple;`
  - Static assertions
What next? [5]

- Stroustrup: C++ is the most successful traditional systems programming language.
- Stroustrup predicts a C++1x standard will come after C++0x.
- Next technical report (TR2) expected after C++0x.
References I

The annotated C++ reference manual. 

Referenced on 6.10.2009.

What is object-oriented programming? 

ACM SIGPLAN Notices, 28(3), Mar 1996.


C++. 

C++0x. 