History

1967  Simula 67

1972-80 Smalltalk

1979  Stroustrup starts 'C with Classes'

1983  'C with Classes' renamed  C++

1985  First commercial release of C++

1986  Objective C

1989  C++ version 2.0 released
     Multiple inheritance, abstact classes static member functions
     const member functions, protected member functions

Goals of C++

Efficient and portable as C

Support multiple programming styles

Give the programmer choice

Compatible with C

Zero-overhead principle

Does not need sophisticated programming environment
Standards

1998
ISO/IEC JTC1/SC22/WG21 working group published C++98

2003
Corrected version of standard ISO/IEC 14882:2003

2005
Library Technical Report 1 (TR1)
Extensions to C++ library
C++0x

New standard for C++

Might be finalized in 2009

Some Goals of C++0x

Maintain compatibility with C++98
Introduction of new features through the standard library
Increase type safety
Increase performance and the ability to work directly with hardware;
Provide proper solutions for real world problems;
Implement “zero-overhead” principle
Make C++ easy to teach and to learn

http://en.wikipedia.org/wiki/C%2B%2B0x
Actually I made up the term “object-oriented”, and I can tell you I did not have C++ in mind.
```cpp
#include <iostream>

int main()
{
    std::cout << "Hello World\n";
}
```

Al pro 1->g++ HelloWorld.cc
Al pro 2->a.out
Hello World
Al pro 3->
Issues

Main
Namespaces
Compiler Directives
IO
Main Arguments

#include <iostream>
using namespace std;

int main(int argc, char *argv[], char *envp[], char *apple[]) {
    cout << "Number of Arguments " << argc << endl;
    for (int index = 0; index < argc; index++) {
        cout << "Argument " << argv[index] << endl;
    }
    cout << "Environment " << envp[0] << endl;
    cout << "OS X extra " << apple[0] << endl;
    return 1;
}

Output

Number of Arguments 1
Argument /Users/whitney/Courses/520/Fall09/C++/HelloWorld.cc.out
Environment TM_SELECTED_FILE=/Users/whitney/Courses/520/Fall09/C++/HelloWorld.cc
OS X extra /Users/whitney/Courses/520/Fall09/C++/HelloWorld.cc.out

Third argument (envp) Only in Microsoft Visual C++, Mac OS X and Darwin
Fourth argument (apple) Mac OS X and Darwin
Namespaces
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello World\n";
}

#include <iostream>
using std::cout;

int main()
{
    cout << "Hello World\n";
}
#Declaring namespaces

```cpp
#include <iostream>
using namespace std;

namespace foo {
  const int count = 1;
}

namespace bar {
  const int count = 2;
}

int main() {
  cout << "foo:count " << foo::count << endl;
  cout << "bar:count " << bar::count << endl;
}
```

Output

```
foo:count 1
bar:count 2
```
# Aliasing namespaces

```cpp
#include <iostream>
using namespace std;

namespace foo {
    const int count = 1;
}

namespace bar {
    const int count = 2;
}

namespace f = foo;

int main() {
    cout << "foo:count " << f::count << endl;
    cout << "bar:count " << bar::count << endl;
}
```
Compiler Directives
# Preprocessor Directives

<table>
<thead>
<tr>
<th>Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define</td>
<td>Define a preprocessor macro</td>
</tr>
<tr>
<td>#undef</td>
<td>Remove a macro definition</td>
</tr>
<tr>
<td>#include</td>
<td>Insert text from another file</td>
</tr>
<tr>
<td>#if</td>
<td>Conditionally include some text, based on the value of a constant expression</td>
</tr>
<tr>
<td>#ifdef</td>
<td>Conditionally include text if a macro name is defined</td>
</tr>
<tr>
<td>ifndef</td>
<td>Conditionally include text if a macro name is not defined</td>
</tr>
<tr>
<td>#else</td>
<td>else clause for if, ifdef, ifndef</td>
</tr>
<tr>
<td>#elif</td>
<td>else if clause</td>
</tr>
</tbody>
</table>
includes

#include <fileName>

Include standard file with fileName
look in standard location for file /usr/include
contains type definitions, function declarations

#include "my_File"

Include user defined file
# Define Example

## File me.h
```c
#define Me
int Roger = 5;
float Whitney = 10.1;
```

## File TestMe.cc
```c
#include <iostream>

#include "me.h"

#ifndef Me
#define Me
#define Me
    int Roger = 10;
    float Whitney = 20.1;
#endif

int main()
{
    std::cout << "Roger's value is: " << Roger << "\n";
}
```

**Output**
Roger's value is: 5
C++ #include verses Java import

#include
includes the given source file into current file

import
Tells compiler where to look for Java class names
Allows you to use short name of a class
More like using
## Macro Names

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LINE</strong></td>
<td>current line number in source code</td>
</tr>
<tr>
<td><strong>FILE</strong></td>
<td>Name of source file</td>
</tr>
<tr>
<td><strong>DATE</strong></td>
<td>Date compile started</td>
</tr>
<tr>
<td><strong>TIME</strong></td>
<td>Time compile started</td>
</tr>
<tr>
<td>__cplusplus</td>
<td>If fully compliant &gt;= 199711L</td>
</tr>
</tbody>
</table>
Sample

```cpp
#include <iostream>
using namespace std;

int main()
{
    cout << "Line number " << __LINE__ << " of file " << __FILE__ << ".\n";
    cout << "Compilation began " << __DATE__ << " at " << __TIME__ << "\n";
    cout << "The compiler's __cplusplus is" << __cplusplus;
    return 0;
}
```
Example from http://www.cplusplus.com/doc/tutorial/preprocessor/
Some IO
cin & cout

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    int a;
    float b;
    char name[50];

    cout << "Please enter an integer, a float, and a string\n";
    cin >> a >> b >> name;

    cout << "Your input: ";
    cout << setw(8) << a << setw(8) << b << setw(8) << name << endl;
}
```

Al pro 31->g++ io.cc
Al pro 32->a.out
Please enter an integer, a float, and a string
12 3.4 cat
Your input: 12 3.4 cat
IO Streams

- **cin**
  - standard input

- **cout**
  - standard output (buffered)

- **cerr**
  - standard error

- **clog**
  - standard error, but buffered
cin Problem

#include <iostream>
using namespace std;

int main()
{
    int tryThisOut;
    for (int K = 0; K < 8; K++)
    {
        cin >> tryThisOut;
        cout << tryThisOut << " ";
    }
}

Input
1 2 3 4.5 6 7 8 9

Output
1 2 3 4 4 4 4 4
Basic Data Type and Statements
Type Sizes and Ranges

Depends on compiler and hardware

```cpp
#include <iostream>
using namespace std;

int main() {

    short int K;
    cout << "sizeof(int): " << sizeof(int) << endl
         << "sizeof(K): " << sizeof(K) << endl
         << "sizeof(float): " << sizeof(float) << endl
         << "sizeof(double): " << sizeof(double) << endl
         << "sizeof(char): " << sizeof(char) << endl;

    return 1;
}
```

sizeof returns the number of bytes. char is always 1 byte. However a byte can vary depending on compiler and the hardware. Normally it is 8 bits, but at least one C++ compiler uses 64 bit bytes. See http://www.parashift.com/c++-faq-lite/intrinsic-types.html#faq-26.4
# C++ Data Types

<table>
<thead>
<tr>
<th>Types</th>
<th>Modifiers</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>signed</td>
<td>short, short int</td>
</tr>
<tr>
<td>int</td>
<td>unsigned</td>
<td>short int, long int</td>
</tr>
<tr>
<td>float</td>
<td>short</td>
<td>unsigned long int</td>
</tr>
<tr>
<td>double</td>
<td>long</td>
<td>long double, long long</td>
</tr>
<tr>
<td>char</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wchar_t</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If a type only has a modifier it is an int.
C++ Basic Statements

if (expression) statement
switch
while
for
do statement while (expression)
break
continue
goto
There are some interesting things

```cpp
#include <iostream>
using namespace std;

int main() {
    long int y = 1;
    int size;
    int x = (y == 1) ? size = sizeof(y), size + 1 : 0;
    cout << x;
    return 1;
}
```
Variables, Pointers, Arrays, References & Const
Variables

#include <iostream>

char a = 'a';

int main()
{
    int b = 10;
    b = b + 5;
    float c;
    std::cout << c << std::endl;

    for (unsigned int d = 2; d > 0; d = d -1)
        std::cout << d << std::endl;
}
Pointers

```cpp
#include <iostream>

int main()
{
    int* ip; // ip is an integer pointer
    int x = 10;

    ip = &x; // ip now points to x
    *ip = 5; // x now equals 5

    std::cout << "ip is: " << ip << "\n";
    std::cout << "*ip is: " << *ip << "\n";
    std::cout << "x is: " << x << "\n";
}
```

Output

```
ip is: 0xbfffe878
*ip is: 5
x is: 5
```
Declaring Pointers

```c
int* ip;       // ip is a pointer to an integer
int *x;        // x is a pointer to an integer
int *x, y, *z; // x and z are pointers to int, y is int
int* x, y, z;  // x is pointer to int, y and z are ints
```
Illegal Pointer Usage

```
int i = 5;
int* ip ;

*ip = &i;    // assigning address to integer

int x = 10;
int y;
int *xp;

*xp = x;    // What address does xp point to?
            // Bad news

xp = &y;
*xp = x;    // This is ok
            // y now equals 10
```
Pointer Fun

```c
int* xp1;
int** xp2;
int*** xp3;
int x = 10;

xp1 = &x;
xp2 = &xp1;
xp3 = &xp2;
***xp3 = 5;  // x now equals 5

int* ip;
char* cp, them = 'w';
int me = 10;

ip = &me;
cp = &them;

*ip = *ip + 2;  // add 2 to me
ip = ip + 2;    // add 2*4 bytes to address in ip

cp = cp + 2;    // add 2 bytes to address in cp
```
Arrays & Pointers

An array name is a pointer to its first element

```c
int ia[5] = { 0, 1, 2, 3, 4};
int *ip = ia;
```

Some true statements

- `ia[0] == *ia`
- `ia[3] == *(ia + 3)`
- `ia[0] == *ip`

Can't change `ia`

- `ia = ia + 1`  // illegal, compiler error
- `ip = ip + 2;`
- `ia[2] == *ip`
Strings, Pointers, and Arrays

char *st = "Hi Mom";  // string of length 7
                     // strings end in \0
int me = 10;
int *ip = &me;

cout << st << "\n";  // prints Hi Mom
cout << *st << "\n";  // prints H
cout << ip << "\n";   // prints 0xf7fff904
cout << *ip << "\n";  // prints 10

cout << *(st + 4) << "\n"; // prints o

st = st + 3;
cout << st << "\n";     // prints Mom
cout << *st << "\n";    // prints M
Strings, Pointers, and Arrays

```cpp
char ca1[] = {'H', 'i', ' ', 'M', 'o', 'm'}; // 6 elements in ca1
char ca2[] = "Hi Mom"; // 7 elements in ca2

cout << ca1 << "\n"; // prints Hi Mom
cout << ca2 << "\n"; // prints Hi Mom
cout << ca1[0] << "\n"; // prints H
cout << ca2[0] << "\n"; // prints H

int ia[] = {1, 2, 3, 4};
cout << ia << "\n"; // prints 0xf7fff8d8
```
main()
{
    float value = 10.1;
    float &firstReferenceValue = value;
    float& secondReferenceValue = value;

    secondReferenceValue = 12.0;
    // value now equals 12.0
}

secondReferenceValue, firstReferenceValue and value are different names for same address
#include <iostream>
using namespace std;

void Rswap( int &v1, int &v2)
{
    int temp = v2;
    v2 = v1;
    v1 = temp;
}

int main()
{
    int a = 10, b = 20;
    Rswap( a, b);
    cout << a << endl << b << endl;
}
#define BUFSIZE 512 /* C style */

main()
{
    const int bufSize = 512;

    bufSize = 1024;  // static error - can't change constant
}

Constants
Const Issues

const double pi; // error: uninitialized const

const int testMe = 5; // constant integer

int* p = &testMe; // error

const int* pc; // pointer to constant integer

pc = &testMe; // ok

int trouble;

pc = &trouble; // ok

trouble = 10; // ok

*pc = 5; // error
# Const Issues

const char* pc = "hi mom"; // pointer to constant
pc[3] = 'a';          // error
pc = "hi dad"         // ok

char *const cp = "Hi Dad"; // constant pointer
cp[3] = 'a';          // ok
cp = "Send Money";   // error

const char *const cpc = "Cat"; // constant pointer to a constant
cp[3] = 'a';          // error
cp = "Send Money";   // error
Const Issues

```c
int MyFunction(const double* trouble)
{
    *trouble = 5.0; // error
}
```