

CS 520 Advanced Programming Languages
Fall Semester, 2009
Doc 6 C++ Intro
Sept 21, 2009

Copyright ©, All rights reserved. 2009 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (<http://www.opencontent.org/opl.shtml>) license defines the copyright on this document.

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, abstract 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

Alan Kay on C++

Actually I made up the term “object-oriented”,
and I can tell you I did not have C++ in mind.

HelloWorld.cc

```
#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

Namespaces

using

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

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

```
#include <iostream>
using std::cout;
```

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

Declaring namespaces

```
#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

```
#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

<code>#define</code>	Define a preprocessor macro
<code>#undef</code>	Remove a macro definition
<code>#include</code>	Insert text from another file
<code>#if</code>	Conditionally include some text, based on the value of a constant expression
<code>#ifdef</code>	Conditionally include text if a macro name is defined
<code>#ifndef</code>	Conditionally include text if a macro name is not defined
<code>#else</code>	else clause for if, ifdef, ifndef
<code>#elif</code>	else if clause

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

```
#define Me  
int Roger = 5;  
float Whitney = 10.1;
```

File TestMe.cc

```
#include <iostream>  
  
#include "me.h"  
  
#ifndef 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

<code>__LINE__</code>	current line number in source code
<code>__FILE__</code>	Name of source file
<code>__DATE__</code>	Date compile started
<code>__TIME__</code>	Time compile started
<code>__cplusplus</code>	If fully compliant \geq 199711L

Sample

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

int main()
{
    cout << "Line number " << __LINE__;
    cout << " of file " << __FILE__ << ".\n";
    cout << "Compilation began " << __DATE__;
    cout << " at " << __TIME__ << ".\n";
    cout << "The compiler's __cplusplus is" << __cplusplus;
    return 0;
}
```

Some IO

cin & cout

io.cc

```
#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

```
#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(int): 4
sizeof(K): 2
sizeof(float): 4
sizeof(double): 8
sizeof(char): 1
```

C++ Data Types

Types

void
int
float
double
char
bool
wchar_t

Modifiers

signed
unsigned
short
long

Examples

short
short int
long int
unsigned long int
long double
long long

C++ Basic Statements

if (expression) statement

switch

while

for

do statement while (expression)

break

continue

goto

There are some interesting things

```
#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

```
#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
```

*

Declares a pointer
Dereference operator

&

Address-of operator

Declaring Pointers

```
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

```
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

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

Some true statements

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

```
ip = ip + 2;  
ia[2] == *ip  
ia[3] == ip[1]
```

Can't change ia

```
ia = ia + 1    // illegal, compiler error
```

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

```
st = st + 3;
cout << st << "\n";         // prints Mom
cout << *st << "\n";       // prints M
```

Strings, Pointers, and Arrays

```
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
```

Reference Types

```
main()
{
    float    value = 10.1;
    float    &firstReferenceValue = value;
    float&   secondReferenceValue = value;

    secondReferenceValue = 12.0;
    // value now equals 12.0
}
```

Pass by Reference

```
#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;
}
```

Output

20

10

Constants

```
#define BUFSIZE 512 /* C style */  
  
main()  
{  
    const int bufSize = 512;  
  
    bufSize = 1024; // static error - can't change constant  
}
```

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
cpc[3] = 'a';                  // error
cpc = "Send Money";           // error
```

Const Issues

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