COMP322 - Introduction to C++

Winter 2011

Lecture 1 - Introduction

Milena Scaccia

School of Computer Science
McGill University

January 4, 2011

Course Facts

- 1-credit crash course in C++
- An overview of the fundamentals of the C++ programming language
- Not an introductory programming course
- 13 Classes, Tuesdays 14:35-15:25, ENGTR 1090
- Course Webpage: `http://www.cs.mcgill.ca/~mscacc/comp322`

Course Instructor

- Milena Scaccia
- Email: `mscacc@cs.mcgill.ca`
- Office: ENGMC 229
- Office hours: Tuesday 13:00-14:00

Teaching Assistants

- Sevan Hanssian
  Email: `shanss@cs.mcgill.ca`
  Office: ENGMC 229
  Office hours: Wednesday 12:00 - 13:00
- Zineng Yuan
  Email: `zineng.yuan@mail.mcgill.ca`
  Office: ENGTR 3103
  Office hours: Friday 10:00 - 11:00
- Yancheng Xiao
  Email: TBA
  Office: TBA
  Office hours: TBA

Assessment

- Two in-class tests, 25% each
- Two homework assignments, 25% each
  - Programming problems
  - 4 weeks per assignment
  - 10% per day late penalty, for up to three days
  - Use GNU C++ ("g++")
  - Homework will be graded based on correctness, style and comments
  - Submitted via *myCourses* `http://www.mcgill.ca/mycourses`
- Academic Integrity: See `http://www.mcgill.ca/integrity`

Calendar

1. 04 Jan - Course introduction
2. 11 Jan - Basic language features (A1 out)
3. 18 Jan - Pointers and references
4. 25 Jan - Memory management
5. 01 Feb - Input/output using the Standard Library
6. 08 Feb - Classes (A1 due)
7. 15 Feb - Test 1

- 22 Feb - Study Break

8. 01 Mar - Operator and function overloading
9. 08 Mar - Inheritance (A2 out)
10. 15 Mar - Exceptions
11. 22 Mar - Templates and STL
12. 29 Mar - Test 2
13. 05 Apr - Optional Topic (A2 due)

Historical Note

http://www2.research.att.com/∼bs/homepage.html

- Begun in 1979 by Bjarne Stroustrup at Bell Labs
- Originally called "C with Classes", but renamed C++in 1983
- "Middle-level" language
- Descendant of C, ancestor of Java

Design principles

- Compiles to machine (binary) code
- Compile-time type checking
- Flexible programming styles
- Low runtime overhead
- Minimal development environment
- Mostly compatible with C

Differences from C

- Classes
- Overloading
- Templates
- Exceptions
- Namespaces

Differences from Java

- Compiles to machine code
- Multiple inheritance
- Pointers and references
- Templates
- No garbage collection

Pros and cons

- Pros:
  - Like C, C++ is useful for systems programming
  - Commercially important!
  - Faster; permits a lower and finer level of control (both a pro and con)
- Criticisms:
  - Allows serious errors and security problems (e.g does not check array indices or initialization; does not check whether a pointer points to an object that no longer exists)
  - Not quite as standard as either C or Java
  - Lots of "missing features", e.g. no multithreading support (although there is a planned new standard for C++ (C++0x) which will address this matter)
  - Can seem complex and difficult

C++ Standard Library

- Collection of common classes and functions
- Includes most of the C Standard Library
- Derived from Standard Template Library (STL)
- Data types: Strings, complex numbers, etc.
- Containers: Lists, sets, queues, stacks, etc.
- Algorithms: Sorting and searching

C++ basics

- Statements terminated with semicolon `;`
- Comments either between `/* .. */` or after `//`
- Basic constants and types largely borrowed from C
- Most operators identical to those in C
- Parentheses are used to group expressions: `a * (b + c)`
- All identifiers must be declared before use, e.g.
  `int count; float average = 0.0;`

C++ basics - Basic types

The sizes and specific range values are typical for 32-bit systems.

Type	Bytes	Min	Max
bool	1	false	true
signed char	1	SCHAR_MIN (-128)	SCHAR_MAX (127)
unsigned char	1	0	UCHAR_MAX (255)
char	1	CHAR_MIN	CHAR_MAX
short [int]	2	SHRT_MIN (-32768)	SHRT_MAX (32767)
unsigned short [int]	2	0	USHRT_MAX (65535)
int	4	INT_MIN	INT_MAX
unsigned [int]	4	0	UINT_MAX
long [int]	4	LONG_MIN	LONG_MAX
unsigned long [int]	4	0	ULONG_MAX
float	4	-FLT_MAX	+FLT_MAX
double	8	-DBL_MAX	+DBL_MAX
long double	8	-LDBL_MAX	+LDBL_MAX

C++ example - hello.cpp

```
#include <iostream>
int main()
{
  std::cout << "Hello, world!\n";
  return 0;  // Return code for success
}
```

This text, contained in the file hello.cpp, is the canonical trivial program, intended to print a friendly greeting.

C++ example - hello.cpp

```
#include <iostream>
int main()
{
  std::cout << "Hello, world!\n";
  return 0;  // Return code for success
}
```

- “#include” is a preprocessor directive
  - Preprocessor runs before the compiler
  - The entire file “iostream” is incorporated
  - No semicolon used in preprocessor statements
  - Incorporates part of standard library

C++ example - hello.cpp

```
#include <iostream>
int main()
{
  std::cout << "Hello, world!\n";
  return 0;  // Return code for success
}
```

- “main()” is a special function
  - Control starts with this function
  - It must be a global function returning `int`
  - Must be defined only once per project
  - Is *not* part of any class

C++ example - hello.cpp

```
#include <iostream>
int main()
{
  std::cout << "Hello, world!\n";
  return 0;  // Return code for success
}
```

- `std::cout` refers to a global object
  - It is an object of the class `ostream`
  - It is similar to the `stdout` global from C
  - The '<<' operator writes the object
  - The '::' is the scope operator

C++ example - hello.cpp

```
#include <iostream>
int main()
{
  std::cout << "Hello, world!\n";
  return 0;  // Return code for success
}
```

- `return` specifies value of function `main()`
  - Takes an (optional) value
  - The number zero is an integer constant
  - In this case, zero indicates success
  - Returns control to calling function

C++ example - Compiling and running

```
$ g++ -Wall -o hw hello.cpp
$ ./hw
Hello, world!
$
```

If you do not have a g++ compiler:

- Linux: Install using the command: **sudo apt-get install g++**
- Windows: Obtain a g++ compiler by installing Cygwin **http://www.cygwin.com/**. Cygwin is a Linux-like environment that runs on top of Windows, which includes the g++ compiler.
- Mac: **http://www.edparrish.com/common/macgpp.php**
- Note that g++ is installed on all machines in the Trottier labs.

C++ basics - Arithmetic operators

```
+       // Addition and unary plus
-       // Subtraction and unary negation
*       // Multiplication
/       // Division
%       // Integer remainder
```

Another important operator is the assignment operator:

```
=       // Assignment
```

Where possible, C++ will automatically convert among the basic types. It is more liberal then Java in accepting code without casting.

C++ basics - Comparison operators

The result of a comparison operator is always a value of type 'bool':

```
==      // equal
!=      // not equal
>       // greater than
<       // less than
>=      // greater than or equal
<=      // less than or equal
```

C++ basics - Logical operators

The logical `&&` and `||` operators use short-circuit evaluation. They execute the right hand argument only if necessary to determine the overall value.

```
&&    // logical and
||    // logical or
!     // logical negation
```

C++ basics - Bitwise operators

These operators support logical operations on bits.

```
&     // bitwise and
|     // bitwise or
^     // bitwise exclusive or
~     // bitwise complement
<<    // left shift
>>    // right shift
```

E.g. In Microcontrollers where available RAM is very limited, we can use bitwise exclusive or to swap two variables of the same type without using a temporary variable:

```
a = a^b;
b = a^b;
a = a^b;
```

C++ basics - if statement

```
// Simplest form
if (response == 'y') return true;

// Less simple
if (result > 0.0) {
  x = 1.0 / result;
  y += x;
}
else {
  std::cout << "Division by zero!";
}
```

C++ basics - `switch` statement

```
int response;

std::cin >> response; // Get input

switch (response) {
case 'y':
    return true;
case 'n':
    return false;
case 'q':
    exit(0);
default:
    std::cout << "I didn't get that, sorry\n";
    break;
}
```

C++ basics - `while` statement

```
int i = 0;
while (i < 10) {
    std::cout << "All work and no play makes
                   Jack a dull boy.\n";
    i++;
}
```

C++ basics - `for` statement

Typically a shorthand for common forms of the `while` statement.

```
for (int i = 0; i < 10; i++) {
   std::cout << "All work and no play makes
                 Jack a dull boy.\n";
}
```

C++ basics - do while statement

```
int i = 0;
do {
   std::cout << "All work and no play makes
                 Jack a dull boy.\n";
   i++;
} while (i < 10);
```

C++ basics - Identifier scope

```
int v = 1;      // Global scope

int main()
{
  int c = 5;  // Local scope

  // Declare 'i' in statement scope
  for (int i = 0; i < c; i++) {
     // do something
  }
  // 'i' is now undefined
  c = c + v;
}
```

C++ basics - Functions

```
/* Addition */
int addition(int x, int y)
{
  int z;
  z = x + y;
  return z;
}
/* Calculate the sum of an array */
double total(double data[], int length)
{
  double sum = 0.0;  // Initialization
  for (int i = 0; i < length; i++)
    sum += data[i];
  return sum;
}
```