CMPT 166: Object-Oriented Programming, using C++

7 January 2009
CMPT166
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

What's on for today

- Languages: machine, assembly, high-level
- C++ compilers and IDEs
- A first C++ program
- Comments and doc-comments
- Compiling and running a C++ program

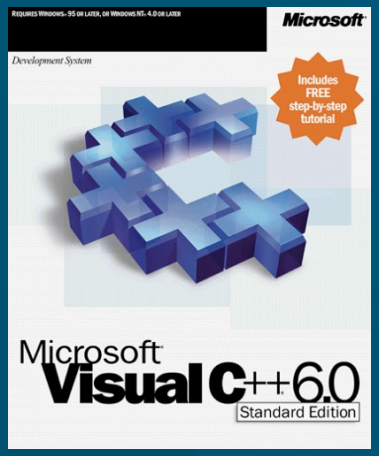
Review: Languages

- Machine language
  - "Native tongue" of computer (CPU, etc.)
  - Highly specific to machine (P4, Atom, ...)
- Assembly language
  - English-like abbreviations for operations
- High-level language
  - More "English-like" instructions
    - Common operations: arithmetic, I/O, etc.
  - Compiler converts to machine language
- Interpreter: execute high-level w/o compile

Compiling

- Python is an interpreted language:
  - When you press F5 in IDLE to run,
  - Code is first compiled into bytecode, then
  - Executed by the Python virtual machine
    - Bytecode: ***.pyc** files (e.g., compile libs)
- C++ is a compiled language:
  - C++ compiler produces object files (*.o)
  - Linked together with libraries
  - To produce executable (*.exe)

C++ compilers



- A few popular C++ compilers:
- Microsoft Visual C++ 2008:
  - Installed on senior lab PCs
  - Industry standard, but expensive
- GNU GCC/G++ 4.3:
  - Installed on senior lab PCs w/Cygwin
  - Free, can install on your own computer
- Intel C++ Compiler (ICC):
  - Well-optimized for Intel CPUs, commercial

Development environments

- Source code is just plain-text, all we need is a text editor and the compiler
- But integrated development environments (IDEs) make life easier
  - IDLE for Python is a basic one
  - MS Visual Studio 2008 is a very complex and expensive one
  - Eclipse is also sophisticated, and free

- Manage multiple projects, classes, and files
- Syntax highlighting, indent, auto-complete

Class policy on IDEs

- For class purposes, you are free to use any development environment you feel comfortable with, **but**:
- I have to be able to re-compile and run your code!
  - Many incompatibilities between compilers
  - I will be using Visual Studio 2008, same as installed on senior lab PCs
  - I also have G++ 4.3.2 on my laptop
- The only officially-supported setup is Visual Studio / Visual C++ in the senior lab

Object-oriented principles

- Alan Kay's Smalltalk (1980): very pure OO
  - C++ is not so pure: OO bolted-on to C
- Five basic principles:
  - Everything is an object: attribs, methods
  - A program is a set of objects passing messages
  - Each object has its own memory, storing other objects
  - Every object has a type (class)
  - All objects of a type can receive the same messages

C++ is object-oriented

- Everything is an object
  - Objects are instances of classes
- Write your program by defining classes
  - Attributes (variables; data)
  - Methods (behaviour; functions)
  - Interfaces (collections of methods)
    - A class may implement more than one interface
    - An interface may be implemented by more than one class

A first C++ program

- #include reads in a library header file
  - Akin to Python 'import'
  - #: C pre-processor directive, not comment!
- using namespace: otherwise would have to prefix cout and endl with 'std::' namespace

```
#include <iostream>
using namespace std;
int main() {
  cout << "Hello, World! I am "
    << 8 << " Today!" << endl;
}
```

Comments and doc-comments

- Comments can either be
  - old-style: /* hi there! */
  - new-style: // hi there!
- Doc-comments start with /** (note two stars)
  - Structured comments can be interpreted by doxygen
  - Similar to Python docstrings
  - @keywords: e.g., @author, @copyright
  - Pre/post-conditions: @param, @return

Compile and run: G++

- Compile: g++ HelloWorld.cpp -o HelloWorld
- Run: ./HelloWorld

- In Visual Studio or Eclipse:
  more complicated, but once you get the project set up, press F5 to compile and run
  (VS: Ctrl-F5 to run without debugging)