Getting Started with C++

9 January 2009
CMPT166
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

Review of last time

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

C++ builtin types

- bool: true, false (note case)
  - Implicit int->bool conversion: anything non-zero converts to true
- char: single character
  - No built-in string, but stay tuned
- int: short int, int, long int
- float: float, double, long double
- Generally, int/double are 32-bits, shorts are 16, etc.
- But on a 64-bit machine, all might be 64-bits

Declarations: vars, functions

- Remember C++ is statically typed:
- All variables and functions need to be declared
  - Variables: type (incl. arrays, pointers, etc.)
    - **int numApples;**
  - Functions: param types, return type
    - **float double_me(float x);**
- Initializing variables:
    - **int numApples = 5;**
- Defining functions:
    - **float double_me(float x) { return x+x; }**

Expression/assignment compatibility

- Statically typed: must declare and initialize variables
  - **int numApples = 5;**
- Cannot assign mismatched types:
  - **numApples = 3.4; // won't work!**
- But values can be promoted to higher-precision
  - **float appleSize;**
  - **appleSize = 3; // from int to float**
    - byte -> short -> int -> long -> float -> double
- Type casting forces a type conversion:
  - **numApples = (int) 3.99; // truncated to 3**

Coding style conventions

- Class names are nouns in CamelCase
- Method names are usually verbs in lowercase:
  - useLowerCamelCase() or use_underscores()
- Local variable names are also lowercase
- Constants: ALL_UPPERCASE

- But! Lots of code is inconsistent; every project has its own conventions

Headers vs. code (impl.)

- A C++ code file may define many variables, functions, and classes
- Its corresponding header file lists declarations so that other files may use them
- mylibrary.cpp:
  - **int numApples = 5;**
  - **int double_me(int x) { return x+x; }**
- mylibrary.h:
  - **extern int numApples;**
  - **int double_me(int x);**

#include headers

- mylibtest.cpp (our main program):
  - **#include "mylibrary.h"**
  - **int main() {**
    - **cout << numApples**
  - **}**
- The #include directive tells the preprocessor to pull in the contents of the file “mylibrary.h”
- Use angle brackets <> to pull standard library headers:
  - **#include <iostream>**

Other preprocessor directives

- #define *NAME text*
  - Preprocessor macro; expands NAME to "text" in your source code
    - **#define PI 3.14159**
- #if *bool_expr*
  - Boolean expression, can use preprocessor macros but *not* your C++ variables
    - **#if PI > 3**
- #ifdef *NAME*
- #endif