CSE 303
Concepts and Tools for Software Development

Magdalena Balazinska

Winter 2010

Lecture 17 –Introduction to C++

Introduction to C++

- Object-oriented language like Java
- Based on C, manual memory management like in C
- Improves many features of C
  - C++ can be used solely as an “improved C” (without defining any classes)
- More complete standard library than C
- The “Standard Template Library” (STL)
  - A lot like Java “collections classes”
  - But not quite the same... so we will discuss them

Plan for This Week

- We will learn just enough C++ to get you started
- Today: the basics
  - Defining and using a simple class
- Friday: memory management
  - When objects are created and destroyed
  - Passing objects by value or by reference
- Monday: inheritance
- Wednesday: templates and STL

Hello World in C++

```
// Include header file from std library
// Note: “new style” header files have no .h
#include <iostream>

int main() {
// Use standard output stream cout
// and operator << to send “Hello World”
// and an end line to stdout
  std::cout << "Hello World" << std::endl;
  return 0;
}
```

C++ Formatted Input/Output

- C++ I/O occurs in streams of bytes
- Stream insertion operator
  - Left shift operator (<<) designates stream output
  - Sends data from a variable to a stream
- Stream extraction operator
  - Right shift operator (>>) designates stream input
  - Extracts data from a stream into a variable
  - Example: `cin >> my_integer;`
- `cout`, `cin`, and `cerr` are stream objects
  - They are connected to `stdout, stdin` and `stderr`

Compiling C++ Programs

- It is standard for C files to have extension .c
- For C++, you can use: `.cpp`, `.cxx`, `.C`, **`.cc`**
- To compile C++ code, use g++ instead of gcc
- Standard example: “Hello World” (`hello.cc`)

```
g++ -Wall -o hello hello.cc
```

- Notes
  - In C++, there are no constraints on filenames
  - You can also put multiple classes in one file

Namespaces

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

int main() {

  cout << "Hello World" << endl;
  return 0;

}
```

Namespaces

- A namespace allows us to group declarations under one name
- Namespaces help avoid name collisions and redefinition errors
- All the elements of the standard C++ library are declared within namespace `std`
- You should always use a namespace for your own declarations

Namespaces

```
#include <iostream>
using namespace std;
namespace MYSPACE {
  typedef struct {
    int a;
  } A;
}
int main() {
  MYSPACE::A sa;
  sa.a = 3;
  cout << sa.a << endl; // Prints: 3
  return 0;
}
```

Our First C++ Class

- Ok... now that we understand “Hello World”, we can get into the heart of things...

- We will examine a class called `Property`
  - We will point out differences between C++ and C
  - As well as difference between C++ and Java
- We will also discuss memory management

A Simple C++ Class

- Examine the `Property` class
  - Class definition in `.h` file
    - Includes member function declarations
    - Can also include function definitions (not recommended)
  - Member function definitions are in `.cc` file
  - Pay close attention to the constructor & destructor
  - Note the access specifiers: `public, private`
  - Note that we can use pointer `this` (in `toString`)
  - How the `static` attribute is declared and initialized
  - The use of namespaces

Member Access Specifiers

- They determine the type of access
  - public: accessible to everyone
  - private: accessible only to member functions
- The access specifiers can appear
  - In any order inside the header file
  - Multiple times, but preferably only once
- Default access mode is private

Function Overloading

- C++ enables function overloading where
  - Several functions have the same name
  - But different parameters
- The compiler selects the appropriate function
  - Matches arguments with parameters
- Examples:
  - The two: `adjustPrice` methods
  - The two constructors

Readings

- For more information, you can read one of many C++ tutorials
  - http://www.cplusplus.com/doc/tutorial/