CS 101: Computer Programming and Utilization

Jan-Apr 2017

Sunita Sarawagi
(cs101@cse.iitb.ac.in)

Lecture 17: Program Organization and Functions

About These Slides

- Based on Chapter 11 of the book

  An Introduction to Programming Through C++

  by Abhiram Ranade (Tata McGraw Hill, 2014)

- Original slides by Abhiram Ranade
  - First update by Sunita Sarawagi

A different role for functions

- We said that a function should be created if you find yourself writing code to perform the same action at different places in the program.
- However, functions have a different role too: A function is an "organizational/logical unit" of a program.

Physical units of code: files

- If several people write different functions of the same program, it is more convenient if each uses a different file.
- We need ways by which functions in one file can call functions in other files

Outline

Functions and program organization

- The main program is a function
- How to split a program into many files
  - Function declarations
  - Separate compilation
  - Header files
- Namespaces
- Using C++ without simplecpp

Splitting a program into many files

- A program may contain several functions. All need not be placed in the same file.
- If code in file F calls a function f, then function f must be declared inside F, textually before any call to it.
- A function definition is a declaration, but there can be other ways to declare.
- Every function must be defined in just one of the files that are used for a program.

Function declaration

- A function declaration is the definition without the body.
  - The return type, name, parameter types and optionally parameter names.
- Example: declaration of gcd function:

```
int gcd(int m, int n);
int gcd(int, int);    // also acceptable.
```

- The declaration tells the compiler that if `gcd` appears later, it will be a function and take 2 ints as arguments.
  - This helps the compiler to translate your program into machine language, without needing to look up the definition of `gcd`.
- If a file calls a function but contains only a declaration of it; it cannot be completely compiled to enable execution.
  - Whatever is in it, is compiled, and the result is called an object module.
  - To get an executable programs, all the object modules containing all called functions must be linked together.

Separate compilation

- File gcd.cpp

```
int gcd(int m, int n){ … }
```

- File lcm.cpp

```
int gcd(int, int);
int lcm(int m, int n){
    return m*n/gcd(m,n);}
```

- File main.cpp

```
int lcm(int, int);
int main(){
cout << lcm(36,24) << endl;
}
```

- function definitions
- function declarations
- As you can see, each file contains a declaration of the function that is called in it.
- You may compile and link all files together by giving

s++ main.cpp lcm.cpp gcd.cpp

- You may compile each file separately, e.g. by giving

s++ -c main.cpp

- -c will ask compiler to produce main.o (object module).
- Object modules can be linked together to get an executable by typing

s++ main.o lcm.o gcd.o

Header files

- Tedious to remember what declaration to include in each file.
- Instead, put all declarations in a header file, and "include" the header file into every file.
- Header files have suffix .h or .hpp., or no suffix.
- The directive "`#include filename`" is used to include files. It is simply replaced by the content of the named file.
- OK to declare functions that do not get used.
- OK to have both a declaration and then the definition of a function in the same file.
- If header file is mentioned in " ", it is picked up from the current directory.
- If it is mentioned in < >, it is picked up from some standard place, e.g. `simplecpp`

- File gcdlcm.h

```
int gcd(int, int);
int lcm(int,int);
```

- File gcd.cpp

```
#include "gcdlcm.h"
int gcd(int m, int n){ … }
```

- File lcm.cpp

```
#include "gcdlcm.h"
int lcm(int m, int n){ … }
```

- File main.cpp

```
#include <simplecpp>
#include "gcdlcm.h"
int main(){
cout << lcm(36,24) << endl;
}
```

Header files for classes

- Typically, separate file for each large class with the same name.
- Header file declares the entire class but skips definition of large functions that are declared in a .cpp file
- Includes similar to other header files.

- **File queue.hpp**

```
class Queue {
   private:
     // declare private data members.
   public:
      // declare, not define large functions
      bool insert(int driver);
      ...
 }
```

- **File queue.cpp**

```
   bool Queue::insert(int driver) {...}
   ...
```

- **File main.cpp**

```
#include "queue.hpp"
int main() {
  Queue q; .... }
```

Concluding Remarks

- Functions are building blocks of programs.
- Functions can be put into many files, provided each file contains a declaration before the use.
- Declarations go into header files.
- Details discussed in the book.