Function Scope and Random Number Generator

CS135 Lab 3

Chang Jia
chang@cse.unr.edu

Feb 12 2008

Assignment 3

- Need to design and implement two complete C++ programs
  - Golf
  - DDR Score
- Fully understand the problems first!
  - Design the program before coding.
- Consider all requirements for the program when designing.
  - **No global variables!**

Assignment 3 – Golf

- Keep the same code as given in golf_main.cpp
  - Global variables
  - Function prototypes
  - Main function
- Test Cases (Two)
  - Any case you choose
  - Tiger Woods case
    - Convert: 270 kilometers per hour => ? meters per second

Review: Predefined Function

Types of Functions

TYPES	**OBJECTIVES IN THE LAB**
Predefined Functions	Learn what they are and how to use them
User-defined Functions	Learn how to define them and use them properly in a program

Predefined Functions

- C++ provides several predefined mathematical functions. These functions are available in the library file cmath (math.h).
- Requires the following include statement

  #include <cmath>

  or #include <math.h>
- Included math functions: http://mathbits.com/mathbits/compsci/LibraryFunc/Math.htm

Simple Example

- Let us consider the following program which uses the sqrt function, and the pow function, both available in the header file cmath: receive an positive number from user, compute and display the square of this number and the square root of this number.

```
#include<iostream>
#include <cmath>   // include cmath library here
using namespace std;

int main ()
{
        double num, root, square;
        cout << "Please enter a positive number: ";
        cin >> num;
        square = pow (num, 2.0);    // function call
        cout << "square of " << num << " is " << square << endl;
        root = sqrt(num);     // function call
        cout << "square root of " << num << " is " << root<<
endl;
        return 0;
}
```

Function Scope

Scope

The **scope** of an identifier can be thought of as the portion of a program in which that identifier has been defined.

- Variables can be **global**, where the scope of the variable is the whole program
- Variables can be **local**, where the scope of the variable is simply the block (or function) in which it is defined

Scope

- A **global variable**, is one that is declared outside any function.
- A **local variable** is declared within the body of a function.

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

int value;      // value is a global variable
void print_out_value ();
int main()
{
    int value2, value3; // local to main
    cin >> value;
    print_out_value;
    //... processing on value, value2, value3
    return 0;
}

void print_out_value ()
{
        int value4, value5; // local to print_out_value
        // ... processing on value, value4, value5
        cout <<"value is " << value;
}
```

(* See details in Global_Local_Variables.cpp)

Random Number Generator

Math Review

- We call an interval of the form (a,b) open; [a,b] closed; [a,b) or (a,b] half-open or half-closed.
- Here are some examples:
  - (3,5) is the set of all numbers greater than 3 and less than 5.
  - (2,6] is the set of all numbers greater than 2 and less than or equal to 6
  - [-1,1] is the set of all numbers greater than or equal to -1 and less than or equal to 1.

Random Number Generator

- Header files - stdlib.h and time.h
- Seed the generator
  - srand (time (NULL));
- Call random number generator
  - int num;
  - num = rand();

Random Number Generator

int num;
num = rand(); // [0, max_int]

What if we want numbers in the range [0,4]?

num=rand()% ? ;

generates a random number between 0 and 4, both included, as [0, 4]

Answer: num = rand()%5;

Random Number Generator

x=rand()%?; //[0, 4]

If rand() returns 0	0%5 = 0
If rand() returns 1	1%5 = 1
If rand() returns 2	2%5 = 2
If rand() returns 3	3%5 = 3
If rand() returns 4	4%5 = 4
If rand() returns 5	5%5 = 0

Random Number Generator

- Seed the random number generator with the srand() function.
  - Can seed with any number or certain number

    srand (time (NULL)) or srand(1)

- Generate a random number we use the rand() function.
  - To generate numbers in the range [0, n]:

    x = rand()% (n +1)

Srand() and Rand() Functions

- To generate numbers in the range [min, max]:

x = rand() % (max – min + 1) + min

Sample Code:

```
#include <stdlib.h>  //required for srand () and rand () functions
#include <time.h>   //required for time () function.

. . .
srand( time (NULL) );   // Initialize random number generator.
. . .
num = rand () % 100 + 1; //Assigning the randomly generated
                         //number to variable num.
```

num needs to be within the range of [1, 100]

(* See the complete code in RandomNumber.cpp)

Rand() Function

- To generate numbers in the range (min, max]:
  - Consider the range (min, max] is equivalent to [min+1, max]:

x = rand() % (max – (min + 1) + 1) + (min + 1)

- To generate numbers in the range [min, max):
  - Consider the range [min, max) is equivalent to [min, max-1]:

x = rand() % ((max – 1) – min + 1) + min

- To generate numbers in the range (min, max):
  - Consider the range (min, max) is equivalent to [min+1, max-1]:

x = rand() % ((max – 1) – (min + 1) + 1) + (min + 1)

Assignment 3 – DDR Score

- If possible, learn how to design the problem from the given executable files.

  http://www.cse.unr.edu/~chang/08Labs.html

- How many times of random number generator need to be called to get 4 numbers?
  - Answer: Need to call srand() once before calling rand(), and call rand() 4 times to get 4 numbers!

Reminders

- Call srand() function prior to the first call of rand() function. Otherwise, the generated number will eventually repeat.
- Don't make the mistake of calling srand() every time you generate a random number; we only usually need to call srand() once.
- Remember the random number generated by

  x = rand() % (max – min + 1) + min

  falling in a closed interval as [min, max]. Figure out the right range before use rand() function!

Reminders

- Ask if you have problems you cannot solve by yourself
  - Email your source code & question
  - Come to Office Hours
- Compile your programs for sure!
  - Compile the programs till no error (better for no warning as well).
- Source code – (Hardcopy & Softcopy)
  - Soft copy should be the same as hard copy.
- No global variables!
- Test cases – (Hardcopy)