Fundamental Computer Programming- C++ Lab(I)

LAB 7
Craps with functions

Week 7, Fall 2022

Rung-Bin Lin

International Bachelor Program in Informatics
College of Informatics
Yuan Ze University

Purposes

- Learn earn how to use functions
- Learn how to measure the runtime of a program execution or a segment of code
- Learn how to use enumeration type
- Learn how to use random number generators

Functions

- **Function definitions**

```
return-value-type function-name( parameter-list )
{
    declarations and statements
}
```

- **Use of functions to develop structured programs.**
  - **Function prototype (function declaration)**

```
return-value-type function-name( parameter-list );
```

    - ✓ **Name**
    - ✓ **Parameters**
    - ✓ **Return type**
  - **Scope rules (concept of local variables)**
  - **Call a function**

```
function-name( argument-list );
```

  - **Argument coercion, i.e., matching between arguments in function call and parameters in function definition**
  - **return from a function**

return *expression;*

```
1  // Fig. 5.3: fig05_03.cpp
2  // Creating and using a programmer-defined function.
3  #include <iostream>
4  using namespace std;
5
6  int square( int ); // function prototype
7                          // Located outside the main function
8  int main()
9  {
10    // loop 10 times and calculate and output the
11    // square of x each time
12    for ( int x = 1; x <= 10; x++ )
13       cout << square( x ) << "  ";  // function call
14
15    cout << endl;
16 } // end main
17
18 // square function definition returns square of an integer
19 int square( int y )  // y is a copy of argument to function
20 {
21    return y * y;      // returns square of y as an int
22 } // end function square
```

```
1  4  9  16  25  36  49  64  81  100
```

Fig. 5.3 | Programmer-defined function `square`.

Argument Coercion

- **Argument values that do not correspond precisely to the parameter types in the function prototype can be converted by the compiler to the proper type before the function is called.**
  - For example, `square(4.5)` returns 16, not `20.25`.
- **These conversions occur as specified by C++'s promotion rules.**

Argument

Lose information

Parameter

Data types	
`long double`	
`double`	
`float`	
`unsigned long int`	(synonymous with `unsigned long`)
`long int`	(synonymous with `long`)
`unsigned int`	(synonymous with `unsigned`)
`int`	
`unsigned short int`	(synonymous with `unsigned short`)
`short int`	(synonymous with `short`)
`unsigned char`	
`char`	
`bool`	

Parameter

Keep information

Argument

Fig. 5.5 | Promotion hierarchy for fundamental data types.

Examples of Argument Coercion

- **Information lost**

```
Int square(int y)
{
Return y*y;
}

Int main( ) {
double x=2.25;
Cout << square(x) << end;
}
```

- **Information kept**

```
Int square( double y)
{
Return y*y;
}

Int main( ) {
int x=2;
Cout << square(x) << end;
}
```

Enumeration Type

enum Status {CONTINUE, WON, LOST};

Status gameStatus;

- **Status now becomes a type. It can be used to define a variable. For example, gameStatus is now a variable of type Status. It can have one of the following three values**
  - ✓ **CONTINUE, assigned a value of 0**
  - ✓ **WON, assigned a value of 1**
  - ✓ **LOST, assigned a value of 2**

Built-in Pseudo Random Number Generator

```
#include <cstdlib>
```

```
rand();
```

- A random number generator. The values generated by this function between 0 and RAND_MAX.

```
srand(time(0));  // set up a seed
```

- time(0) is a function that returns current time in seconds. The return value will be used as a seed for srand(…) function.

Runtime Measurement

```
#include <ctime>
        .
        .
        .
clock_t t1,t2;
t1=clock();
// place the most time consuming part of your code here, i.e., between
// t1=clock() and t2=clock.
// There should be no statement of using cin to get data from keyboard
t2=clock();
double diff=(double)t2-(double)t1;
cout << “The runtime of my program = “ << diff/CLOCKS_PER_SEC << endl;
```

Problem Description of Craps

A player rolls two dice. Each die has six faces. These faces contain 1, 2, 3, 4, 5 and 6 spots. After the dice have come to rest, the sum of the spots on the two upward faces is calculated. If the sum is 7 or 11 on the first roll, the player wins. If the sum is 2, 3 or 12 on the first roll (called "craps"), the player loses (i.e., the "house" wins). If the sum is 4, 5, 6, 8, 9 or 10 on the first roll, then that sum becomes the player's "point." To win, you must continue rolling the dice until you "make your point." The player loses by rolling a 7 before making the point.

The program in Fig. 5.10 simulates the game. In the rules, notice that the player must roll two dice on the first roll and on all subsequent rolls. We define function `rollDice` (lines 63–75) to roll the dice and compute and

Original Code of Craps Game (1)

```
// Fig. 5.10: fig05_10.cpp
// Craps simulation.
#include <iostream>
#include <cstdlib> // contains prototypes for functions srand and rand
#include <ctime> // contains prototype for function time
using namespace std;

int rollDice(); // rolls dice, calculates and displays sum

int main()
{
   // enumeration with constants that represent the game status
   enum Status { CONTINUE, WON, LOST }; // all caps in constants

   int myPoint; // point if no win or loss on first roll
   Status gameStatus; // can contain CONTINUE, WON or LOST
|
```

Original Code of Craps Game (2)

```
// randomize random number generator using current time
srand( time( 0 ) );

int sumOfDice = rollDice(); // first roll of the dice

// determine game status and point (if needed) based on first roll
switch ( sumOfDice )
{
   case 7: // win with 7 on first roll
   case 11: // win with 11 on first roll
      gameStatus = WON;
      break;
   case 2: // lose with 2 on first roll
   case 3: // lose with 3 on first roll
   case 12: // lose with 12 on first roll
      gameStatus = LOST;
      break;
   default: // did not win or lose, so remember point
      gameStatus = CONTINUE; // game is not over
      myPoint = sumOfDice; // remember the point
      cout << "Point is " << myPoint << endl;
      break; // optional at end of switch
} // end switch
```

Original Code of Craps Game (3)

```
// while game is not complete
while ( gameStatus == CONTINUE ) // not WON or LOST
{
   sumOfDice = rollDice(); // roll dice again

   // determine game status
   if ( sumOfDice == myPoint ) // win by making point
      gameStatus = WON;
   else
      if ( sumOfDice == 7 ) // lose by rolling 7 before point
         gameStatus = LOST;
} // end while

// display won or lost message
if ( gameStatus == WON )
   cout << "Player wins" << endl;
else
   cout << "Player loses" << endl;
} // end main
```

Original Code of Craps Game (4)

```
// roll dice, calculate sum and display results
int rollDice()
{
   // pick random die values
   int die1 = 1 + rand() % 6; // first die roll
   int die2 = 1 + rand() % 6; // second die roll

   int sum = die1 + die2; // compute sum of die values

   // display results of this roll
   cout << "Player rolled " << die1 << " + " << die2
      << " = " << sum << endl;
   return sum; // end function rollDice
} // end function rollDice
```

Output of Original Craps Game (5)

```
Player rolled 5 + 1 = 6
Point is 6
Player rolled 5 + 6 = 11
Player rolled 5 + 6 = 11
Player rolled 2 + 5 = 7
Player loses
```

```
Player rolled 6 + 4 = 10
Point is 10
Player rolled 3 + 4 = 7
Player loses
```

```
Player rolled 1 + 3 = 4
Point is 4
Player rolled 1 + 6 = 7
Player loses
```

LAB 7: Craps with functions

- **Modify the program Crap simulation given in Fig. 5.10 in the textbook "C++: How to Program" as follows:**
  - **Rewrite the main function in Fig. 5.10 into a function**
    - ✓ **The function prototype should be Status crapsFunc();**
    - ✓ **The return value should be either "WON" or "LOST".**
    - ✓ **The game rule is now modified as follows:**
      If the sum is one of the numbers in A={ some numbers from 2 to 12} on the first roll, the player wins. If the sum is one of the numbers in B={some numbers from 2 to 12} on the first roll (called craps), the player loses. If the sum is one of the number in C={some numbers from 2 to 12} on the first roll, then the sum becomes the player's point. To win you must continue rolling the dice until you make your point. The player loses by rolling a number contained in D and D $\subset$ A.
      The sets of A, B, and C should satisfy the following rules: $A \cap B = \varnothing$, $B \cap C = \varnothing$, $A \cap C = \varnothing$, and $A \cup B \cup C = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$. A, B, C, and D are non-empty.
    - ✓ **You should determine the numbers in the sets A, B, C, and D such that the player's win probability is as close to 0.57 as possible.**

Lab 7 cont1.

- ✓ **You should still use rollDice() function to roll dices.**
- ✓ **You should delete lines 55~59 and lines 71~73**
- ✓ **You should not include line 19 in this function. Include it into the new main() function discussed below.**
- ✓ **(Optional +20% if done) Within the rollDice(), you can replace the pseudo random number generator rand() by the LFSR pseudo random number generator you designed in Lab 6. The length of the LFSR should be 32. The seed of the LFSR in each run should be initially set to time(0). This LFSR pseudo random number generator should be implemented as a function called randLFSR(…); You should give a proper parameter list and a return-value type for this function. You can refer to page 5 of https://www.xilinx.com/support/documentation/application_notes/xapp052.pdf to determine the tap bit positions of your LFSR.**

Lab 7 cont2.

- **Write a new main() function that reads an integer which is the number of times the Craps game will be played. The win probability should be as close to 0.57 as possible.**
  - **Print out a prompting message “ Enter the number of times of Craps game to be played:” to get the number of times the Craps game being played at each run.**
  - **Print out the win probability in each run and the runtime for each run. Refer to the example input and output for more prompting messages. The x in [Rx] for an output prompting message is the run number. For example, [R1] refers to the prompting message for the output of first run**
  - **Repeat playing the game many runs until the player would like to stop playing. Print out the probability that is closest to 0.57 among all the runs you made.**
  - **Calculate your grade for the Lab using the following formula:**

    100-abs(1.0-best WIN probability/0.57)*1000

For example, if the best WIN probability is 0.56933, your score will be 100-abs(1.0-0.56933/0.57)*1000 = 100-abs(1.0-99882…)*1000=98.8.

The grade should be calculated on the premise that you play the game 20 runs and the number of craps game in each run is 200000.

Example of Input & Output

- **The win probability should be different for each run.**

```
    Enter the number of times of Craps games to be played: 200000
[R1] Win probability = 0.570485  Time elapsed: 2 seconds
    Continue to play? (Y or y for yes): y
[R2] Win probability = 0.572855  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R3] Win probability = 0.5691  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R4] Win probability = 0.57374  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R5] Win probability = 0.57099  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R6] Win probability = 0.57368  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R7] Win probability = 0.571965  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R8] Win probability = 0.57215  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R9] Win probability = 0.571005  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R10] Win probability = 0.572985  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R11] Win probability = 0.57006  Time elapsed: 2 seconds
    Continue to play? (Y or y for yes): y
[R12] Win probability = 0.57283  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R13] Win probability = 0.56974  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R14] Win probability = 0.57124  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R15] Win probability = 0.570935  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R16] Win probability = 0.57071  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R17] Win probability = 0.57189  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R18] Win probability = 0.571105  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R19] Win probability = 0.5713  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): y
[R20] Win probability = 0.5722  Time elapsed: 1 seconds
    Continue to play? (Y or y for yes): n
[OUT] The best win probability = 0.57006 obtained at R11
[OUT} Score = 99
```

Grading Policy

- **Your grade of the Lab will be calculated when the number of craps game in each is 200000 and you play it 20 runs.**
- **The grade will be equal to**

  100-abs(1.0-best WIN probability/0.57)*1000

Here, abs() is a function calculating the absolute value of a given number.

Grading Rules for TA

- Have to check whether the original main() function is rewritten into a function Status crapsFunc().
- Must check whether randLFSR(int) is used to generate pseudo random numbers.
- Have to check whether the win probability of the player is close to 0.57
- Have to check whether the game can be replayed many times as shown in the output example.
- Must check that the best win probability appears among the plays just made.
- Must check whether runtime of each run is printed.