ITE 365 Lab04: Don't play Craps!

Adam Lewis

<2015-05-18 Mon>

Contents

1 Objectives

1. Work with arrays in C#
2. Continue our discussion with random numbers

2 Background

In lecture, we looked at how to use the random number generator in C# to simulate playing Craps, a game of chance. In this lab, we want to consider whether or not one should actually participate in this endeavor while gambling in the casino.

In this lab, we will write an app that runs 1000 games of craps and answers the following questions:

1. How many games are won on each of the first 20 rolls and every roll afterwards?
2. How many games are lost on the each of the same rolls?
3. What are the changes of wining at craps?
4. What is the average length of a game of craps?

This is an excellent application of using the array data structure.

3 Instructions

Start a new C# Console application in C#. Name your app as *CrapsSimulationApp* and name your class as being *CrapsSimulationClass*.

3.1 Setup the data structures needed by the simulation

Let's reuse some of the data structures we defined in our Craps simulation from the lecture:

```
// create random number generator for use in method RollDice
private static Random randomNumbers = new Random();

// enumeration with constants that represent the game status
private enum Status { CONTINUE, WON, LOST };

// enumeration with constants that represent common rolls of the dice
private enum DiceNames
{
   SNAKE_EYES = 2,
   TREY = 3,
   SEVEN = 7,
   YO_LEVEN = 11,
   BOX_CARS = 12
}
```

Now let's define the arrays and accumulation variables we will be using to keep track of the data for answering our questions about the game:

```
static int[] wins; // number of wins, by rolls
static int[] losses; // number of losses, by rolls
static int winSum = 0; // total number of wins
static int loseSum = 0; // total number of losses
```

3.2 Creating the simulation

Let us add the structure of the simulation into our **main()** application:

```
public static void Main( string[] args )
{
   int sumOfDice = 0; // sum of the dice
   int myPoint = 0; // point if no win or loss on first roll

   Status gameStatus; // can contain CONTINUE, WON or LOST

   int roll; // number of rolls for the current game

   wins = new int[ 22 ]; // frequency of wins
   losses = new int[ 22 ]; // frequency of losses

   for ( int i = 1; i <= 1000; i++ )
   {
     // Play the game
   } // end for

   PrintStats();
} // end Main
```

The first portion of the game play is the first roll. This code comes from what we did in the lecture:

```
sumOfDice = RollDice(); // first roll of the dice
roll = 1;

// determine game status and point based on sumOfDice
switch ( ( DiceNames ) sumOfDice )
{
   case DiceNames.SEVEN: // win with 7 on first roll
   case DiceNames.YO_LEVEN: // win with 11 on first roll
```

```
      gameStatus = Status.WON;
      break;
   case DiceNames.SNAKE_EYES: // lose with 2 on first roll
   case DiceNames.TREY: // lose with 3 on first roll
   case DiceNames.BOX_CARS: // lose with 12 on first roll
      gameStatus = Status.LOST;
      break;
   default: // did not win or lose, so remember point
      gameStatus = Status.CONTINUE; // game is not over
      myPoint = sumOfDice; // remember the point
      Console.WriteLine( "Point is {0}", myPoint );
      break;
} // end switch
```

If we don't have a winner/loser on the first roll, then we need to keep rolling until we hit a winner. We adapt the code from the lecture to keep track of the number of rolls.

```
// while game is not complete ...
while ( gameStatus == Status.CONTINUE )
{
   sumOfDice = RollDice(); // roll dice again
   ++roll;

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

// all roll results after 20th roll placed in last element
if ( roll > 21 )
   roll = 21;

// increment number of wins in that roll
if ( gameStatus == Status.WON )
{
   ++wins[ roll ];
   ++winSum;
} // end if
else // increment number of losses in that roll
{
   ++losses[ roll ];
   ++loseSum;
} // end else
```

And at the end of all of this, don't forget to display the stats:

```
PrintStats();
```

3.3 Now to actually print the stats

For display, we use the formatting capabilities of the **WriteLine** method to display the stats we collected. We also need to generate the summary stats we want to create:

```
// display win/loss statistics
public static void PrintStats()
{
   int totalGames = winSum + loseSum; // total number of games
   int length = 0; // total length of the games

   // display number of wins and losses on all rolls
   for ( int i = 1; i <= 21; i++ )
   {
      if ( i == 21 )
         Console.WriteLine( "{0} {1} {2} {3}",
            wins[ i ], "games won and", losses[ i ],
            "games lost on rolls after the 20th roll" );
      else
         Console.WriteLine( "{0} {1} {2} {3}{4}",
            wins[ i ], "games won and", losses[ i ],
            "games lost on roll #", i );

      // for calculating length of game
      // number of wins/losses on that roll multiplied
      // by the roll number, then add them to length
      length += wins[ i ] * i + losses[ i ] * i;
   } // end for

   // calculate chances of winning
   Console.WriteLine( "\n{0} {1} / {2} = {3:F}%",
      "The chances of winning are", winSum, totalGames,
      ( 100.0 * winSum / totalGames ) );

   Console.WriteLine( "The average game length is {0:F} rolls.",
      ( ( double ) length / totalGames ) );
} // end method PrintStats
```

3.4 And for rolling the dice

For the actual game play, we will reuse the code used in the lecture:

```
// roll dice, calculate sum and display results
 public static int RollDice()
 {
    // pick random die values
    int die1 = randomNumbers.Next( 1, 7 );
    int die2 = randomNumbers.Next( 1, 7 );
    int sum = die1 + die2; // sum die values

    return sum; // return sum of dice
 } // end method RollDice
```

3.5 Build and run the application

Build, debug, and run your application. What opinion do you now have about Craps now that you have run this simulation?

4 Submission instructions

Combine a copy of your source code and a screen-shot of your program into a PDF file (you can use Microsoft Word to do this) and attach it to the assignment on Blackboard.