CC-112 Programming Fundamentals

Random Number Generation

Nazar Khan
Department of Computer Science
University of the Punjab

The `rand()` function

- Function `rand` generates an integer between 0 and RAND_MAX which is defined by the C standard to be at least 32767.
- Values produced by rand can be scaled and shifted to produce values in a specific range.
- The general equation for scaling and shifting a random number is

  ```
  n = a + rand() % b;
  ```

  where `a` is the shifting value (i.e., the first number in the desired range of consecutive integers) and `b` is the scaling factor (i.e,. the width of the desired range of consecutive integers).

The `srand()` function

- To randomize a program, use the C standard library function `srand`.
- The `srand` function seeds the random number generator.
- An `srand` call is ordinarily inserted in a program only after it has been thoroughly debugged.
- While debugging, it's better to omit `srand`.
- This ensures *repeatability*, which is essential to proving that corrections to a random number generation program work properly.
- The function prototypes for `rand` and `srand` are contained in `<stdlib.h>`.
- To randomize without the need for entering a seed each time, we use `srand(time(NULL))`.

Example: A Game of Chance

Rules 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 throw, the player wins. If the sum is 2, 3, or 12 on the first throw (called "craps"), the player loses (i.e., the "house" wins). If the sum is 4, 5, 6, 8, 9, or 10 on the first throw, 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.*

Simulation of “Craps”

```
// Simulating the game of craps.
#include <stdio.h>
#include <stdlib.h>
#include <time.h> // contains prototype for function time

// enumeration constants represent game status
enum Status { CONTINUE, WON, LOST };
int rollDice(void); // function prototype

int main(void)
{
  // randomize random number generator using current time
  srand(time(NULL));

  int myPoint; // player must make this point to win
  enum Status gameStatus; // can contain CONTINUE, WON, or LOST
  int sum = rollDice(); // first roll of the dice

  // determine game status based on sum of dice
  switch(sum) {

    // win on first roll
    case 7: // 7 is a winner
    case 11: // 11 is a winner
      gameStatus = WON;
      break;
```

Simulation of “Craps”

```
    // lose on first roll
    case 2: // 2 is a loser
    case 3: // 3 is a loser
    case 12: // 12 is a loser
      gameStatus = LOST;
      break;

    // remember point
    default:
      gameStatus = CONTINUE; // player should keep rolling
      myPoint = sum; // remember the point
      printf("Point is %d\n", myPoint);
      break; // optional
  }

  // while game not complete
  while (CONTINUE == gameStatus) { // player should keep rolling
    sum = rollDice(); // roll dice again
    // determine game status
    if (sum == myPoint) { // win by making point
      gameStatus = WON;
    }
    else {
      if (7 == sum) { // lose by rolling 7
        gameStatus = LOST;
      }
    }
  }
```

Simulation of “Craps”

```
  // display won or lost message
  if (WON == gameStatus) { // did player win?
    puts("Player wins");
  }
  else { // player lost
    puts("Player loses");
  }
}

// roll dice, calculate sum and display results
int rollDice(void)
{
  int die1 = 1 + (rand() % 6); // pick random die1 value
  int die2 = 1 + (rand() % 6); // pick random die2 value
  // display results of this roll
  printf("Player rolled %d + %d = %d\n", die1, die2, die1 + die2);
  return die1 + die2; // return sum of dice
}
```