Random Number Generation

ESC108A Elements of Computer Science and Engineering
B. Tech. 2017

Course Leaders:

Roopa G.

Ami Rai E.

Chaitra S.

Objectives

- At the end of this lecture, student will be able to
  - explain the necessity of random number generators
  - use pseudo random generation function provided by the C standard library

Contents

- Random Number generation
- Fixing the bounds

Question

- Can you write a program to pick a number between 1 and 20?

Random Number Generation

- Getting a random number

- Consider the following statement

```
int i = rand();
```

➢ The function prototype for rand() is in <stdlib.h>

➢ The value produced directly by rand() are always in the range

0 ≤ rand() ≤ RAND_MAX (constant in <stdlib.h>)

– Standard C states that the value of RAND_MAX must be at least 32767, which is the maximum value for a 16-bit integer

Random Number Generation contd.

- Every number between 0 and RAND_MAX has an equal chance (or probability) of being chosen each time rand is called

- The range of values produced directly by rand is often different from what is needed in a specific application
  - A program that simulates coin tossing might require only 0 for "heads" and 1 for "tails."
  - A dice-rolling program that simulates a six-sided die would require random integers from 1 to 6

Scaling

- Consider a program to simulate 20 rolls of a six-sided die and print the value of each roll

- We use

  rand() % 6

  – to produce integers in the range 0 to 5
- This is called scaling
- The number 6 is called the scaling factor
- We then shift the range of numbers produced by adding 1 to our previous result

  1+rand() % 6

Scaling - Example

- The output of the program confirms that the results are in the range 1 to 6
- The output might vary by compiler

```
int i; /* counter */
for ( i = 1; i <= 20; i++ ) {
    /* pick random number from 1 to 6 and output it */
    printf( "%10d", 1 + ( rand() % 6 ));
    /* if counter is divisible by 5, begin new line of output */
    if ( i % 5 == 0 ) {
        printf( "\n" );
    } /* end if */
} /* end for */
```

Scaling - Generalisation

- The statement to simulate the rolling of a six-sided die

  face = 1 + rand() % 6;

  – always assigns an integer value (at random) to the variable face in the range 1 ≤face ≤6

- We can generalize this result as follows

  n = a + rand() % b;

  ➢ a is the shifting value (which is equal to the first number in the desired range of consecutive integers) and b is the scaling factor (which is equal to the width of the desired range of consecutive integers)

Randomizing

- rand() generates pseudorandom numbers
  - Calling rand() repeatedly produces a sequence of numbers that appears to be random
  - however, the sequence repeats itself each time the program is executed

- Randomizing
  - Once a program has been thoroughly debugged, it can be conditioned to produce a different sequence of random numbers for each execution
- Accomplished with the standard library function srand

srand()

- srand()
  - takes an unsigned integer argument

    void srand(unsigned int seed);

    srand(123200890);
  - seeds function rand to produce a different sequence of random numbers for each execution of the Program

- Setting a Seed
  - Controls the numbers generated
  - same seed generates the same sequence of random numbers
- The function prototype for srand is found in <stdlib.h>

srand() - Example

```
int i;  /* counter */
printf( "Enter seed: " );
scanf( "%u", &seed );  /* note %u for unsigned */
srand( seed );  /* seed random number generator */
/* loop 10 times */
for ( i = 1; i <= 10; i++ ) {
    /* pick a random number from 1 to 6 and output it */
   printf( "%10d", 1 + ( rand() % 6 ) );
   /* if counter is divisible by 5, begin a new line of output */
    if ( i % 5 == 0 ) {
      printf( "\n" );
   } /* end if */
} /* end for */
```

srand() using time

- To randomize without entering a seed each time, use a statement like

  srand(time(NULL));

  - causes the computer to read its clock to obtain the value for the seed automatically
- Function time returns the current time of day in seconds
- This value is converted to an unsigned integer and used as the seed to the random number generator
- Function time takes NULL as an argument
- The function prototype for time is in <time.h>

Summary

- C provides random number generation routines
- rand() can be used when program requires a random choice
- rand() generates pseudorandom numbers

Further Reading

Kernighan, B. W. and Richie, D. (1992) *The C Programming Language.* 2nd ed., New Delhi:PHI.