CS1010 http://www.comp.nus.edu.sg/~cs1010/

Programming Methodology

UNIT 10

Random Numbers

Unit 10: Random Numbers

Objective:

- Learn the use of random numbers

Unit 10: Random Numbers

1. Introduction
2. rand()
3. srand()
4. “Randomising” the Seed
5. The HiLo Game

1. Introduction

- In simulation and games, we need random number generation.
- Computer cannot generate true random numbers. At most, they could generate pseudo-random numbers, close enough to being random and good enough for most practical purposes.
- We will learn two functions here:
  - rand()
  - srand()

2. rand() (1/2)

- Run the following program Unit10_Random1.c

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
  int i;

  for (i = 1; i <= 10); i++)
    printf("%d\n", rand());

  return 0;
}
```

<stdlib.h> needed to use rand() function

Unit10_Random1.c

```
16838
5758
10113
17515
31051
5627
23010
7419
16212
4086
```

The same set of numbers are generated every time the program is run!

2. rand() (2/2)

- In sunfire, rand() generates an integer in the range [0, 32676]. (Note: [a, b] indicates a closed range, i.e. the range is inclusive of both a and b.)
- The same set of numbers are printed every time the program is run because the numbers are picked from a pre-determined sequence based on some seed.
- Question: How to generate an integer in the range [101, 500]?

Unit10_Random2.c

```
for (i = 1; i <= 10); i++)
  printf("%d\n", rand()%400 + 101);
```

In general, to generate an integer in the range [a, b], we write:

`rand()%(b-a+1) + a`

(This is not the best way, but a simple technique for our purpose.)

3. srand() (1/2)

- As mentioned, these “random numbers” generated are the same from run to run, due to the same default seed being used.
- To get a different set of random numbers each time the program is run, the trick is to change the seed, by calling the srand() function.
- A particular seed (which is an integer) indicates which pre-determined sequence of pseudo-numbers to use, and a subsequent call to rand() will pick up the next number from this sequence.
- Hence, you need only call srand() function once, before you call the rand() function.

3. srand() (2/2)

- Test out the program Unit10_Random3.c

Unit10_Random3.c

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
  int seed, i;

  printf("Enter seed: ");
  scanf("%d", &seed);
  srand(seed); // feed srand() with a new seed

  for (i = 1; i <= 10); i++)
    printf("%d\n", rand()%400 + 101);

  return 0;
}
```

```
Enter seed: 3
248
408
466
413
323
297
274
444
493
308
```

```
Enter seed: 27
351
199
284
249
242
449
402
425
351
445
```

4. “Randomising” the Seed (1/2)

- In the preceding example, the user is asked to enter a value for the seed.
- However, in many applications such as games or simulations, we want to “automate” this step since we do not want user’s invention.
- How do we ensure that every time the program is run, a different seed is used?
- One simple solution is to use the time(NULL) function, which returns an integer that is the number of seconds since 1st of January 1970. This value can then be used as the seed for the srand() function.

4. "Randomising" the Seed (2/2)

Unit10_Random4.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void) {
  int i;

  srand(time(NULL));

  for (i = 1; i <= 10); i++)
    printf("%d\n", rand()%400 + 101);

  return 0;
}
```

<time.h> needed to use time() function

```
408
368
136
360
429
474
378
359
120
229
```

```
117
117
388
357
367
242
341
483
300
382
```

5. The HiLo Game (1/3)

- We will illustrate with the HiLo game, where user is asked to guess a secret number between 1 and 100 inclusive, given up to 5 attempts.

```
*** Welcome to the HiLo game! ***

Guess a number between 1 and 100 inclusive.
Enter your guess [1]: 50
Your guess is too low!
Enter your guess [2]: 70
Your guess is too low!
Enter your guess [3]: 90
Your guess is too high!
Enter your guess [4]: 83
Your guess is too high!
Enter your guess [5]: 76
Too bad. The number is 72. Better luck next time!

Do you want to play again (y/n)?
```

5. The HiLo Game (2/3)

Unit10_HiLo.c

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

void play_a_game(int);

int main(void) {
   int  secret;
   char response;

   srand(time(NULL));

   printf("*** Welcome to the HiLo game! ***\n");
   do {
      secret = rand()%100 + 1;
      play_a_game(secret);
      printf("Do you want to play again (y/n)? ");
      scanf(" %c", &response);
   } while (response == 'y');

   printf("\n*** Thanks for playing. Bye! ***\n");

   return 0;
}
```

5. The HiLo Game (3/3)

Unit10_HiLo.c

```
// Play one HiLo game
void play_a_game(int secret) {
   int guess, tries = 0;
   printf("\nGuess a number between 1 and 100 inclusive.\n");
   do {
      tries++;
      printf("Enter your guess [%d]: ", tries);
      scanf("%d", &guess);

      if (guess < secret)
         printf("Your guess is too low!\n");
      else if (guess > secret)
         printf("Your guess is too high!\n");
   } while ( (tries < 5) && (guess != secret) );

   if (guess == secret) {
      printf("Congratulations! You did it in %d step", tries);
      if (tries == 1) printf(".\n");
      else            printf("s.\n");
   }
   else
      printf("Too bad. The number is %d. Better luck next time!\n",
             secret);
}
```

Summary

- In this unit, you have learned about
  - Generating pseudo-random numbers using rand()
  - Seeding a pseudo-random sequence using srand()
  - Providing a “random” seed by using time(NULL) in the srand() function

End of File