# Laboratory 1

Title of the Laboratory Exercise: Data types, local variables and Random number generation

1. Introduction and Purpose of Experiment

Students get familiar with the data types and local variables and random number generation. Basic concepts such as data types and local variables are part and parcel of almost all the c programs. Hence sound knowledge is most essential in this regard. Also, the random number generation essential for many applications, for ex. rolling a dice for many in gaming applications such as backgammon which requires a random number generation from 1 to 6.

1. Aim and Objectives

Aim

To design and develop a C programs using Data types, local variables and Random number generation to demonstrate the use and significate of the same in programming.

Objectives

At the end of this lab, the student will be able to

* + - Use variables of the basic data types with proper declarations
    - Read and validate the input data
    - Generate random numbers for any application

1. Pseudo Code

Get min, max, n

Seed random with current system time

Set i = 0

While i < n

Print rand()%(max - min + 1) + min

Set i = i + 1

Stop

1. Analysis and Discussions

rand() is an library function that is defined in rand.c as

/\* Return a random integer between 0 and RAND\_MAX. \*/

int

rand (void)

{

return (int) \_\_random ();

}

As the source-code of \_\_random from glibc states, it uses Linear Congruential Generator to generate the pseudo-random numbers, to generate a different set of numbers at every run event, the algorithm is seeded with a seeding value, and here we take the system time to do so.

The generator is defined by:

Where

1. Conclusions

A benefit of LCGs is that with appropriate choice of parameters, the period is known and long. Although not the only criterion, too short a period is a fatal flaw in a pseudorandom number generator.

While LCGs are capable of producing pseudorandom numbers which can pass formal tests for randomness, this is extremely sensitive to the choice of the parameters m and a. For example, a = 1 and c = 1 produces a simple modulo-m counter, which has a long period, but is obviously non–random.

C uses LCGs internally inside rand for generating these pseudorandom numbers, this can be verified by looking at the definition written in glibc. This cannot be used for cryptographic purposes at all, due to the fact that two systems producing generating random numbers using this algorithm may generate the same numbers arising conflicts and encryption will fail.

Nevertheless this algorithm still holds good for small applications such as in an embedded systems where memory is severely limited.