EXPERIMENT 1

Demonstrate the use of data types, local variables and Random function. Tabulate the output for various inputs and verify against expected values. Analyze the efficiency of the algorithm. Describe your learning along with the limitations of overall approach if any. Suggest how these can be overcome in conclusion. Write a C program to illustrate random number generation. Your report should include:

1. Introduction and Purpose of Experiment
2. Aim and Objectives
3. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
4. Implement C program
5. Presentation of Results
6. Conclusion

1. Introduction and purpose of the experiment

This experiment familiarizes us with data types and local variables and random number generation. Basic concepts such as data types and local variables are part of all C programs. Hence sound knowledge is most essential in this regard. Also, the random number generation essential for many applications. It finds use in gambling, statistical sampling, simulation, cryptography, completely randomized designs, and other fields where unpredictable outcomes are desired. Hence, having a good grasp on random number generation is valuable for a programmer.

2. Aim and Objectives

To design and develop a C program to illustrate the usage of random number generation.

3. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code

Algorithm:

```
STEP 1: START

STEP 2: Declare variables

            num ← 0

            i ← 0

STEP 3: read num

STEP 4: Repeat steps until i = num

            4.1 display rand()

STEP 5: set seed using srand(time(0))

STEP 6: i ← 0

STEP 7: Repeat steps until i = num

            7.1 display rand()
```

Pseudocode:

```
Initialize num to 0, i to 0

Input num

While i++ < num

      Print rand()

Initialize i = 0

Srand(time(0))

While i++ < num

      Print rand()
```

4. **Implement the C program**

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

int main(int argc, char** argv) {

    int num = 0;
    printf("Enter the number of numbers to generate: ");
    scanf("%d", &num);

    for (int i = 0;i < num;i++){
        printf("the random number generated is: %d\n", rand());
    }

    printf("\n\nUsing seed:\n");
    srand(time(0));

    for(int i = 0; i < num; i++){
        printf("the random number generated using a seed is: %d\n", rand());
    }

    return 0;
}
```

Figure 1: Source Code

5. Presentation of Results

```
cd '/home/ouroboros/Documents/College/DSA-lab/netbeans/tutorial_1_i'
/usr/bin/make -f Makefile CONF=Debug
"/usr/bin/make" -f nbproject/Makefile-Debug.mk QMAKE= SUBPROJECTS= .build-conf
make[1]: Entering directory '/home/ouroboros/Documents/College/DSA-lab/netbeans/tutorial_1_i'
"/usr/bin/make"  -f nbproject/Makefile-Debug.mk dist/Debug/GNU-Linux/tutorial_1_i
make[2]: Entering directory '/home/ouroboros/Documents/College/DSA-lab/netbeans/tutorial_1_i'
make[2]: 'dist/Debug/GNU-Linux/tutorial_1_i' is up to date.
make[2]: Leaving directory '/home/ouroboros/Documents/College/DSA-lab/netbeans/tutorial_1_i'
make[1]: Leaving directory '/home/ouroboros/Documents/College/DSA-lab/netbeans/tutorial_1_i'

BUILD SUCCESSFUL (total time: 4s)
```

Figure 2: Build

```
Enter the number of numbers to generate: 3
the random number generated is: 1804289383
the random number generated is: 846930886
the random number generated is: 1681692777


Using seed:
the random number generated using a seed is: 1547971099
the random number generated using a seed is: 1564091228
the random number generated using a seed is: 1249224785

RUN FINISHED; exit value 0; real time: 1s; user: 0ms; system: 0ms
```

Figure 3: Execution

```
Enter the number of numbers to generate: 3
the random number generated is: 1804289383
the random number generated is: 846930886
the random number generated is: 1681692777


Using seed:
the random number generated using a seed is: 1473508513
the random number generated using a seed is: 951145027
the random number generated using a seed is: 1637891492

RUN FINISHED; exit value 0; real time: 1s; user: 0ms; system: 0ms
```

Figure 4: Execution

6. Conclusion

The usage of the random number generator in C is learnt. The main function which generates the random number is the `rand()` function. This function outputs a random (not completely random, rather a pseudo random number) between 0 and RAND_MAX. RAND_MAX is a constant whose default value may vary between implementations, but it is granted to be atleast 32767. If the rand function is used as is, it gives the same sequence of random numbers regardless of the number of times the program is run. This is because, at its core, the algorithm generating the random number in the `rand()` function performs operation on a number called a 'seed'. Using this seed, the random number is generated.

The reason why the same sequence of numbers was generated lies with the seed. Since the seed does not change with successive runs, the function outputs the same sequence of numbers. To overcome this limitation, it is necessary to 'seed' the generator. Changing the seed ensures that the sequence of random numbers that are generated are unique and don't repeat with successive runs. This is illustrated in the call to `srand()` function, which takes in a seed. The seed given is the current time. This ensures that the same seed won't occur twice, giving successful random number generation.

Even with all this, the numbers generated are not purely random, as they are defined by some algorithm. Hence they are called pseudo random numbers.