

Experiment 2

Generation Random Number Using Linear Congregational Method

1. Objective :

- To Implement Linear Congregational Method in C to Generate a Series of Random Number

2. Theory:

A random number is a number that is unpredictable and does not follow any pattern. In computer science, random numbers are often required for simulations, cryptography, games, and other applications. However, computers are deterministic machines, so they use special mathematical formulas to generate sequences of numbers that appear random. These are called *pseudo-random numbers*. One of the simplest and most commonly used methods to generate pseudo-random numbers is the **Linear Congruential Method (LCM)**.

The formula for the Linear Congruential Method is:

$$X(n+1) = (a * Xn + c) \% m$$

Where:

- X_n = current number (seed)
- $X(n+1)$ = next random number
- a = multiplier
- c = increment
- m = modulus
- $\%$ = modulo operator (gives the remainder after division)

The values of a , c , m , and the initial seed X_0 are chosen carefully to get good random numbers. The numbers generated will repeat after a while (called the period), but for many practical uses, they are good enough to be considered random.

3. Demonstration :

Source code :

```
#include <stdio.h>
void linear(int seed,int a, int incrementer,int modulus){
    int randomNumber;
    for(int i=0;i<5;i++){
        randomNumber=(seed*a + incrementer) % modulus;
        printf("%d. Random No:- %d\n",i+1,randomNumber);
        seed=randomNumber;
    }
}
int main(){
    int seed,a,incrementer,modulus;
    printf("Enter seed:\n");
    scanf("%d",&seed);
    printf("Enter the value of A:\n");
    scanf("%d",&a);
    printf("Enter the incrementer:\n");
    scanf("%d",&incrementer);
    printf("Enter the value of modulus:\n");
    scanf("%d",&modulus);
    linear(seed,a,incrementer,modulus);
    return 0;
```

```
}
```

Output :

```
● csit_5th_sem/snm » ./a.out
Enter seed:
25
Enter the value of A:
3
Enter the incrementer:
61
Enter the value of modulus:
75
1. Random No:- 61
2. Random No:- 19
3. Random No:- 43
4. Random No:- 40
5. Random No:- 31
```

Discussion :

In this program, we implemented the Linear Congruential Method (LCM) to generate pseudo-random numbers. The function linear() takes four parameters:

- seed – the starting value (initial random number),
- a – the multiplier,
- incrementer – the constant increment value,
- modulus – the value used to keep the numbers within a certain range using modulo operation.

The loop runs 5 times to generate and print 5 random numbers. After each number is generated, the seed is updated with the new random number so that the next number depends on the previous one. The user inputs all required parameters at runtime, making this program flexible and allowing testing with different values to observe the output behavior.

4. Conclusion

In this lab, we successfully generated random numbers using the Linear Congruential Method. We learned that although the numbers generated by this method are not truly random, they are good enough for most practical applications. The formula uses a simple mathematical expression with a seed, multiplier, increment, and modulus to generate a sequence of numbers that appear random.

Experiment 3

Generation Random Number Using Multiplicative Congregational Method

1. Objective :

- To Implement Multiplicative Congregational Method in C to Generate a Series of Random Number

2. Theory:

Random numbers are widely used in fields like simulations, games, security, and statistics. Since computers are not capable of generating truly random numbers, they use mathematical formulas to generate pseudo-random numbers—numbers that look random but are actually generated in a predictable way.

One of the common methods for generating such numbers is the Multiplicative Congruential Method (MCM). It is a simplified version of the Linear Congruential Method, and it removes the increment term.

The formula used in this method is:

$$X(n+1) = (a * Xn) \% m$$

Where:

- X_n = current number (seed)
- $X(n+1)$ = next random number
- a = multiplier
- m = modulus
- $\%$ = modulo operator (gives the remainder after division)

This method works well when the values of a , m , and seed are chosen carefully. The generated numbers will repeat after some time, but for many basic applications, they serve the purpose of appearing random.

3. Demonstration :

Source code :

```
#include <stdio.h>
void multiplicative(int seed,int a,int modulus){
    int randomNumber;
    for(int i=0;i<5;i++){
        randomNumber=(seed*a) % modulus;
        printf("%d. Random No:- %d\n",i+1,randomNumber);
        seed=randomNumber;
    }
}
int main(){
    int seed,a,modulus;
    printf("Enter seed:\n");
    scanf("%d",&seed);
    printf("Enter the value of A:\n");
    scanf("%d",&a);
    printf("Enter the value of modulus:\n");
    scanf("%d",&modulus);
    multiplicative(seed,a,modulus);
}
```

```
    return 0;  
}
```

Output :

```
● csit_5th_sem/snm » ./a.out  
Enter seed:  
47  
Enter the value of A:  
7  
Enter the value of modulus:  
100  
1. Random No:- 29  
2. Random No:- 3  
3. Random No:- 21  
4. Random No:- 47  
5. Random No:- 29
```

Discussion :

In this program, we have used the Multiplicative Congruential Method to generate random numbers. The function `multiplicative()` takes three parameters:

- seed (starting value),
- a (multiplier),
- modulus (to keep the values within a range).

Here, unlike the Linear Congruential Method, there is no incrementer, which makes the formula and the code simpler. After each random number is generated, it is assigned as the new seed for the next iteration.

This method is easy to implement and works effectively for basic random number generation when appropriate values are used. However, care must be taken when choosing a seed, and modulus to avoid short repeating cycles.

4. Conclusion

We successfully implemented the Multiplicative Congruential Method for generating pseudo-random numbers. This method is simpler than the Linear Congruential Method and does not use an increment value. It shows how mathematical operations can be used to create sequences that appear random and are useful in various programming applications.

Experiment 4

Generation Random Number Using Mixed Congregational Method

1. Objective :

- To Implement Mixed Congregational Method in C to Generate a Series of Random Number

2. Theory:

Random numbers are essential in many applications like cryptography, simulations, and games. Since computers are deterministic, they can't generate true random numbers. Instead, they use mathematical formulas to create sequences of pseudo-random numbers, which appear random. The Mixed Congruential Method is a popular and improved technique for generating such numbers. It is a type of Linear Congruential Generator (LCG) that combines both multiplication and addition.

The formula is:

$$X(n+1) = (a * Xn + c) \% m$$

Where:

- X_n = current number (seed)
- $X(n+1)$ = next random number
- a = multiplier
- c is the increment (non-zero),
- m = modulus
- $\%$ = modulo operator (gives the remainder after division)

This method is called "mixed" because it uses both a multiplier and a non-zero increment. Compared to the pure multiplicative method, this approach can help generate a longer cycle of numbers and reduce repetition.

3. Demonstration :

Source code :

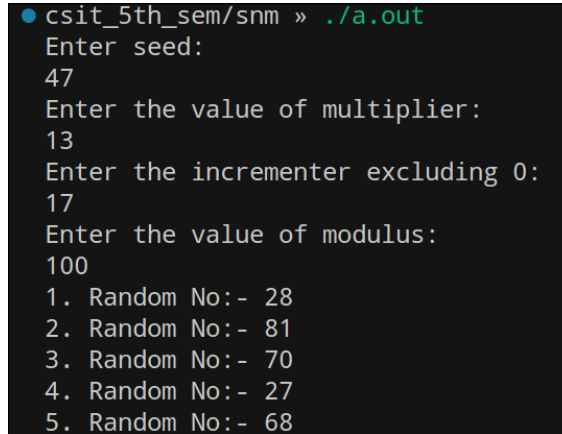
```
#include <stdio.h>
void mixed(int seed,int a, int incrementer,int modulus){
    int randomNumber;
    for(int i=0;i<5;i++){
        randomNumber=(seed*a + incrementer) % modulus;
        printf("%d. Random No:- %d\n",i+1,randomNumber);
        seed=randomNumber;
    }
}
int main(){
    int seed,a,incrementer,modulus;
    printf("Enter seed:\n");
    scanf("%d",&seed);
    printf("Enter the value of multiplier:\n");
    scanf("%d",&a);
    inc:
    printf("Enter the incrementer excluding 0:\n");
    scanf("%d",&incrementer);
```

```

    if(incrementer==0){
        printf("Please enter other value then 0\n\n");
        goto inc;
    }
    printf("Enter the value of modulus:\n");
    scanf("%d",&modulus);
    mixed(seed,a,incrementer,modulus);
    return 0;
}

```

Output :



```

● csit_5th_sem/snm » ./a.out
Enter seed:
47
Enter the value of multiplier:
13
Enter the incrementer excluding 0:
17
Enter the value of modulus:
100
1. Random No:- 28
2. Random No:- 81
3. Random No:- 70
4. Random No:- 27
5. Random No:- 68

```

Discussion :

In this program, we have used the Mixed Congruential Method to generate random numbers. The function `mixed()` takes four parameters:

- seed – the starting value (initial random number),
- a – the multiplier,
- incrementer – the constant increment value,
- modulus – the value used to keep the numbers within a certain range using modulo operation.

The incrementer is checked to ensure it is not zero, as required in the mixed method. If the user enters 0, the program asks again until a valid number is entered. The seed is updated with each generated number so that the next random number depends on the previous one. This gives a better distribution of values and avoids short repeating patterns.

4. Conclusion

We successfully implemented the Mixed Congruential Method to generate random numbers. This method improves upon the multiplicative method by adding a non-zero increment. It helps to generate a wider and more varied sequence of pseudo-random numbers, making it more suitable for real-world applications.