1. **Fibonacci Series in C**

Fibonacci Series in C: In case of fibonacci series, *next number is the sum of previous two numbers* for example 0, 1, 1, 2, 3, 5, 8, 13, 21 etc. The first two numbers of fibonacci series are 0 and 1.

There are two ways to write the fibonacci series program:

- Fibonacci Series without recursion
- Fibonacci Series using recursion

Fibonacci Series in C without recursion

Let's see the fibonacci series program in c without recursion.

```
1. #include<stdio.h>
2. int main()
3. {
4.  int n1=0,n2=1,n3,i,number;
5.  printf("Enter the number of elements:");
6.  scanf("%d",&number);
7.  printf("\n%d %d",n1,n2);//printing 0 and 1
8.  for(i=2;i<number;++i)
    //loop starts from 2 because 0 and 1 are already printed
9.  {
10.         n3=n1+n2;
11.         printf(" %d",n3);
12.         n1=n2;
13.         n2=n3;
14.        }
15.         return 0;
```

```
16.         }
```

Output:

Enter the number of elements: 15

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

Fibonacci Series using recursion in C

Let's see the fibonacci series program in c using recursion.

```
1.  #include<stdio.h>
2.  void printFibonacci(int n){
3.      static int n1=0,n2=1,n3;
4.      if(n>0){
5.          n3 = n1 + n2;
6.          n1 = n2;
7.          n2 = n3;
8.          printf("%d ",n3);
9.          printFibonacci(n-1);
10.             }
11.         }
12.         int main(){
13.             int n;
14.             printf("Enter the number of elements: ");
15.             scanf("%d",&n);
16.             printf("Fibonacci Series: ");
```

```
17.             printf("%d %d ",0,1);
18.             printFibonacci(n-2);//n-
    2 because 2 numbers are already printed
19.          return 0;
20.         }
```

OUTPUT:

Enter the number of elements:15

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

2. Prime Number program in C

Prime number in C: Prime number is a number that is greater than 1 and divided by 1 or itself. In other words, prime numbers can't be divided by other numbers than itself or 1. For example 2, 3, 5, 7, 11, 13, 17, 19, 23.... are the prime numbers.

Note: Zero (0) and 1 are not considered as prime numbers. Two (2) is the only one even prime number because all the numbers can be divided by 2.

Let's see the prime number program in C. In this c program, we will take an input from the user and check whether the number is prime or not.

```
1. #include<stdio.h>
2. int main(){
3. int n,i,m=0,flag=0;
4. printf("Enter the number to check prime:");
5. scanf("%d",&n);
6. m=n/2;
```

```
7. for(i=2;i<=m;i++)
8. {
9. if(n%i==0)
10.     {
11.     printf("Number is not prime");
12.     flag=1;
13.     break;
14.     }
15.     }
16.     if(flag==0)
17.     printf("Number is prime");
18.     return 0;
19.      }
```

Output:

```
Enter the number to check prime:56
Number is not prime

Enter the number to check prime:23
Number is prime
```

3. Palindrome program in C

Palindrome number in c: A **palindrome number** is *a number that is same after reverse*. For example 121, 34543, 343, 131, 48984 are the palindrome numbers.

Palindrome number algorithm

- Get the number from user

- Hold the number in temporary variable
- Reverse the number
- Compare the temporary number with reversed number
- If both numbers are same, print palindrome number
- Else print not palindrome number

Let's see the palindrome program in C. In this c program, we will get an input from the user and check whether number is palindrome or not.

```
#include<stdio.h>
int main()
{
int n,r,sum=0,temp;
printf("enter the number=");
scanf("%d",&n);
temp=n;
while(n>0)
{
    r=n%10;
    sum=(sum*10)+r;
    n=n/10;
    }
    if(temp==sum)
    printf("palindrome number ");
    else
    printf("not palindrome");
    return 0;
    }
```

Output:

```
enter the number=151
palindrome number

enter the number=5621
not palindrome  number
```

4. Factorial Program in C

Factorial Program in C: Factorial of n is the *product of all positive descending integers*. Factorial of *n* is denoted by n!. For example:

1. 5! = 5*4*3*2*1 = 120
2. 3! = 3*2*1 = 6

Here, 5! is pronounced as "5 factorial", it is also called "5 bang" or "5 shriek".

The factorial is normally used in Combinations and Permutations (mathematics).

There are many ways to write the factorial program in c language. Let's see the 2 ways to write the factorial program.

- Factorial Program using loop
- Factorial Program using recursion

Factorial Program using loop

Let's see the factorial Program using loop.

```
1. #include<stdio.h>
2. int main()
3.   {
4.  int i,fact=1,number;
5.  printf("Enter a number: ");
6.   scanf("%d",&number);
7.    for(i=1;i<=number;i++)
8.        {
9.      fact=fact*i;
10.       }
11.       printf("Factorial of %d is: %d",number,fact);
12.      return 0;
13.       }
```

OUTPUT:

```
Enter a number: 5
Factorial of 5 is: 120
```

Factorial Program using recursion in C

Let's see the factorial program in c using recursion.

```
1. #include<stdio.h>
2.
3. long factorial(int n)
4. {
5.   if (n == 0)
6.     return 1;
7.   else
8.     return(n * factorial(n-1));
9. }
10.
11.       void main()
12.       {
13.         int number;
14.         long fact;
15.         printf("Enter a number: ");
16.         scanf("%d", &number);
17.
18.         fact = factorial(number);
19.         printf("Factorial of %d is %ld\n", number, fact);
20.         return 0;
21.       }
```

Output:

```
Enter a number: 6
Factorial of 5 is: 720
```

Armstrong Number in C

Before going to write the c program to check whether the number is Armstrong or not, let's understand what is Armstrong number.

Armstrong number is *a number that is equal to the sum of cubes of its digits*. For example 0, 1, 153, 370, 371 and 407 are the Armstrong numbers.

Let's try to understand why **153** is an Armstrong number.

1. 153 = (1*1*1)+(5*5*5)+(3*3*3)
2. where:
3. (1*1*1)=1
4. (5*5*5)=125
5. (3*3*3)=27
6. So:
7. 1+125+27=153

Let's try to understand why **371** is an Armstrong number.

1. 371 = (3*3*3)+(7*7*7)+(1*1*1)
2. where:
3. (3*3*3)=27
4. (7*7*7)=343
5. (1*1*1)=1
6. So:
7. 27+343+1=371

Let's see the c program to check Armstrong Number in C.

```
1. #include<stdio.h>
2.  int main()
3. {
4. int n,r,sum=0,temp;
5. printf("enter the number=");
6. scanf("%d",&n);
7. temp=n;
8. while(n>0)
9. {
10. r=n%10;
11. sum=sum+(r*r*r);
```

```
12. n=n/10;
13. }
14. if(temp==sum)
15. printf("armstrong  number ");
16. else
17. printf("not armstrong number");
18. return 0;
19. }
```

OUTPUT:

enter the number=153
armstrong number

enter the number=5
not armstrong number

Sum of digits program in C

C program to sum each digit: We can write the sum of digits program in c language by the help of loop and mathematical operation only.

Sum of digits algorithm

To get sum of each digits by c program, use the following algorithm:

- Step 1: Get number by user
- Step 2: Get the modulus/remainder of the number
- Step 3: Sum the remainder of the number
- Step 4: Divide the number by 10
- Step 5: Repeat the step 2 while number is greater than 0.

Let's see the sum of digits program in C.

```
1. #include<stdio.h>
2.  int main()
3. {
4. int n,sum=0,m;
5. printf("Enter a number:");
6. scanf("%d",&n);
7. while(n>0)
8. {
9. m=n%10;
10.        sum=sum+m;
```

```
11.     n=n/10;
12.     }
13.     printf("Sum is=%d",sum);
14.     return 0;
15.     }
```

Output:

```
Enter a number:654
Sum is=15

Enter a number:123
Sum is=6
```

C Array

An array is defined as the collection of similar type of data items stored at contiguous memory locations. Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc. It also has the capability to store the collection of derived data types, such as pointers, structure, etc. The array is the simplest data structure where each data element can be randomly accessed by using its index number.

C array is beneficial if you have to store similar elements. For example, if we want to store the marks of a student in 6 subjects, then we don't need to define different variables for the marks in the different subject. Instead of that, we can define an array which can store the marks in each subject at the contiguous memory locations.

By using the array, we can access the elements easily. Only a few lines of code are required to access the elements of the array.

Properties of Array

The array contains the following properties.

- Each element of an array is of same data type and carries the same size, i.e., int = 4 bytes.
- Elements of the array are stored at contiguous memory locations where the first element is stored at the smallest memory location.
- Elements of the array can be randomly accessed since we can calculate the address of each element of the array with the given base address and the size of the data element.

Advantage of C Array

1) Code Optimization: Less code to the access the data.

2) Ease of traversing: By using the for loop, we can retrieve the elements of an array easily.

3) Ease of sorting: To sort the elements of the array, we need a few lines of code only.

4) Random Access: We can access any element randomly using the array.

Disadvantage of C Array

1) Fixed Size: Whatever size, we define at the time of declaration of the array, we can't exceed the limit. So, it doesn't grow the size dynamically like LinkedList which we will learn later.

Declaration of C Array

We can declare an array in the c language in the following way.

```
data_type array_name[array_size];
```

Now, let us see the example to declare the array.

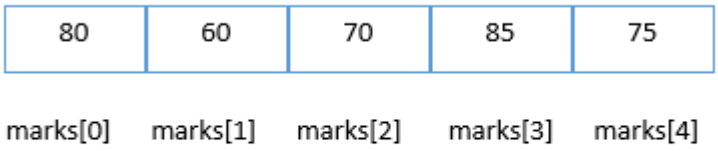
```
int marks[5];
```

Here, int is the *data_type*, marks are the *array_name*, and 5 is the *array_size*.

Initialization of C Array

The simplest way to initialize an array is by using the index of each element. We can initialize each element of the array by using the index. Consider the following example.

```
1. marks[0]=80;//initialization of array
2. marks[1]=60;
3. marks[2]=70;
4. marks[3]=85;
5. marks[4]=75;
```



Initialization of Array

C array example

```
1. #include<stdio.h>
2. int main(){
3. int i=0;
```

```
4. int marks[5];//declaration of array
5. marks[0]=80;//initialization of array
6. marks[1]=60;
7. marks[2]=70;
8. marks[3]=85;
9. marks[4]=75;
10.       //traversal of array
11.       for(i=0;i<5;i++){
12.       printf("%d \n",marks[i]);
13.       }//end of for loop
14.       return 0;
15.       }
```

Output

```
80
60
70
85
75
```

C Array: Declaration with Initialization

We can initialize the c array at the time of declaration. Let's see the code.

```
1. int marks[5]={20,30,40,50,60};
```

In such case, there is **no requirement to define the size**. So it may also be written as the following code.

```
1. int marks[]={20,30,40,50,60};
```

Let's see the C program to declare and initialize the array in C.

```
1. #include<stdio.h>
2. int main(){
3. int i=0;
4. int marks[5]={20,30,40,50,60};//declaration and initialization of array

5.  //traversal of array
6. for(i=0;i<5;i++){
7. printf("%d \n",marks[i]);
```

```
}
return 0;
    }
```

Output

```
20
30
40
50
60
```

C Array Example: Sorting an array

In the following program, we are using bubble sort method to sort the array in ascending order.

```
#include<stdio.h>
void main ()
{
    int i, j,temp;
    int a[10] = { 10, 9, 7, 101, 23, 44, 12, 78, 34, 23};
    for(i = 0; i<10; i++)
    {
        for(j = i+1; j<10; j++)
        {
                if(a[j] > a[i])
                {
                    temp = a[i];
                    a[i] = a[j];
                    a[j] = temp;
                }
            }
        }
        printf("Printing Sorted Element List ...\n");
        for(i = 0; i<10; i++)
        {
            printf("%d\n",a[i]);
        }
```

```
23.       }
```

Program to print the largest and second largest element of the array.

```
1.  #include<stdio.h>
2.  void main ()
3.  {
4.      int arr[100],i,n,largest,sec_largest;
5.      printf("Enter the size of the array?");
6.      scanf("%d",&n);
7.      printf("Enter the elements of the array?");
8.      for(i = 0; i<n; i++)
9.      {
10.             scanf("%d",&arr[i]);
11.         }
12.         largest = arr[0];
13.         sec_largest = arr[1];
14.         for(i=0;i<n;i++)
15.         {
16.            if(arr[i]>largest)
17.            {
18.               sec_largest = largest;
19.               largest = arr[i];
20.            }
21.            else if (arr[i]>sec_largest && arr[i]!=largest)
22.            {
23.               sec_largest=arr[i];
24.            }
25.         }
26.         printf("largest = %d, second largest = %d",largest,sec_largest);

27.
28.       }
```