Chapter - 7
Array

1. Array :- An array in C is a Collection of Similiar data items stored at Contiguous memory locations and elements can be Accessed randomly Using indices of an array. They Can be Used to store collection of primitive data types such as int, float, double, char etc

Declaration of Array :- There are different ways to declare Arrays in Clanguage
To declare an array, we must provide following Information
1. Data type of an array
2. The name of the array
3. The number of subscripts in an array
4. The maximum value of each Subscript

Like other Variables in C, an array has to be declared before it is used. Following is the Syntax for declaring an array

<Datatype> <array name>[n]

Example :-

int student[50]

Declares an array student of integer data type that can store 50 elements. That means, it is equivalent to 50 different variables of Integer type. Each element in the array has a index number which starts from 0 and goes upto n-1 Each member can be addressed as student[0], student[1], student[2] and so on. The

Index number of array always starts from 0 and not from 1.

Intializing Arrays :- We can intialize an array in C either one by one or using a single statements as follows :-

double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};

The number of values between braces cannot be larger than the number of elements that we declare for the array between square brackets [ ]

If you omit the size of array an array just big enough to hold intialization is created.

double balance[] = {1000.0, 2.0, 3.4, 7.0, 50.0};

We can assign a single element of array

balance[4] = 50.0;

Accessing Array :- A single dimensional or linear array is accessed by using a loop by which a variables is used as the index number of array. The index variable intialized to 0 at beginning of the loop and it is incremented to a number, one lesser than the size of array. for example array ar[5] has the elements ar[1], ar[2], ar[3], ar[4]. We can access these elements through a variable 'i' whose value starts from 0 and goes upto 4. So a single statement ar[i] replaces the individual elements

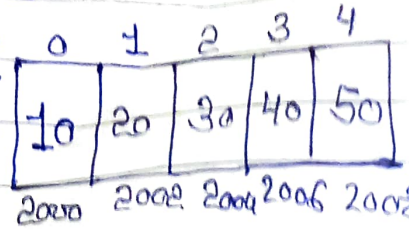
names of array. In each iteration of Loop the value of "i" is incremented and next element of array is accessed.

Program :-

```
#include <stdio.h>
#include <conio.h>
Void main()
{
 int ar[5];
 int i;
  for(i=0;i<5,i++)
      {
        printf("\n Enter array element:");
        scanf("%d",&ar[i]);
      }
 printf("\n The entered values are:");
      for(i=0;i<5;i++)
      {
        printf("%d",ar[i]);
      }
getch();
}.
```

One dimensional array :- Once array is defined the each element of an array can be accessed by Array name followed by an index enclosed in brackets. All array elements are numbered starting from 0

Eg :- int S[5] = {10,20,30,40,50};



2.7 Two dimensional array in C :- The two dimensional array can be defined as an array of arrays. The 2 dimensional array is organized as matrices which can be represented as collection of rows and columns. However 2d arrays are created to implement a relational data base look like data structure. In 2d array each element can be accessed simply by array name followed by two index numbers enclosed in brackets.
Eg : int a[2][3] = {10,20,30,40,50,60}

Declaration of two dimensional Array in C :-
The syntax to declare 2D array is given below

data_type array_name[rows][columns];

Example :-

int twodimen[4][3];

Here 4 is the number of rows and 3 is the number of columns

Intialization of 2D array in C :- In single dimensional array, we don't need to specify the size of array if declaration and intialization are being done simultaneously. However this will not work with 2d arrays we will have to define atleast the second dimension of array. The 2d array can be declared and defined as :

int arr[4][3] = {{1,2,3},{2,3,4}{3,4,5},{4,5,6}};

Example of Two dimensional array :-

```
#include<stdio.h>
#include<conio.h>
Void main()
{
 int a[5][4];
 int i,j;                    //input
 for(i=0;i<5;i++)
 {
    for(j=0;j<4;j++)
    {
      printf("\t Enter the element");
      Scanf("%d",&a[i][j];
    }
 printf("\n");
}
printf("\n You entered \n");   //output
for(i=0;i<5;i++)
 {
    for(j=0;j<4;j++)
    {
      printf("%d \t";j<4;j++)
    }
 printf("\n");
}
getch();
}
```

24 Program to add two matrix into third matrix:-

```
#include <stdio.h>
#include <conio.h>
Void main()
{
  int ar1[4][4] = {
                    {1,2,3,4}
                    {1,2,3,4}
                    {1,2,3,4}
                    {4,2,3,4}
                  };
int ar2[4][4] = {
                    {1,2,3,4}
                    {1,2,3,4}
                    {1,2,3,4}
                    {1,2,3,4}
                  };
int ar3[4][4];
int i,j;
clrscr();
//input
for(i=0;i<4,i++)
{
   for(j=0;j<4;j++)
   {
     ar3[i][j] = ar1[i][j] + ar2[i][j];
   }
}
printf("\n the array after addition is \n");
```

matrix:

```
//output
for(i=0; i<4; i++)
{
  for(j=0; j<4; j++)
  {
    printf("%d \t", ar3[i][j];
  }
  printf("\n");
}
getch();
}
```

Passing an array as parameter to function :- We can pass an entire array as an argument or parameter to a function this can be done by using two methods

1. Passing array as an argument using Subscript notation
2. Passing array using pointer notation

1. Subscript notation method :- In the subscript notation method we pass the array name along with a variable to hold the size of the array. Both of them are used as actual parameters. The similar declaration done for formal parameters. In the function prototype and function definition, a pair of empty square braces follow the name of the array but in the function call, only the names of the array and size is passed.

Example :-

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

```
Void main()
{
 int ar[5], i;
 int max(int tempar[], int size);
   printf("\n Input element in array");
    for(i=0; i<5; i++)
    {
      printf("\n Enter the value");
      scanf("%d", &ar[i]);
    }
   printf("\n The largest element array is %d", max(ar,5));
   getch();
}
 int max(int tempar[], int size)
 {
   int m=0; int i;
   for(i=0; i<size; i++)
   {
     if(tempar[i]>m)
     {
     m = tempar[i];
   }
 }
 return(m);
}
```

Strings in C :- The string can be defined as the one dimensional array of characters terminated by null ['\0']. The character array or string is used to manipulate text such as word or sentences. Each character in the array occupies one byte of memory and last character

must always be 0. The termination character ('\0') is important in a string since it is the only way to identify where the string ends. When we define a string as char S[10], the character S[10] is implicitly initialized with null in memory

Declaration :- There are two ways to declare a string in C language.

1. By char array
2. By string literal

Example :-

char ch[11] = {'J', 'a', 'v', 'a', 'S', 'h', 'a', 'R', 'M', 'A', '\0'};

we know that array index starts from 0, so it will be represented as :-

0	1	2	3	4	5	6	7	8	9	10
J	a	v	a	S	h	a	R	M	A	\0

While declaring string, size is not mandatory. So we can write the above code as

String Literal :-

char ch[] = {"Javasharma"};

char ch[] = {'J', 'a', 'v', 'a', 'S', 'h', 'a', 'R', 'M', 'A', '\0'};

String Example in C :-

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main()
{
char ch[11] = {'J', 'a', 'v', 'a', 'S', 'h', 'a', 'R', 'm', 'a', '\0'};
```

```
char ch2[11] = "javatpoint";
printf("Char Array Value is %s \n", ch);
printf("string literal value is %s \n", ch2);
return 0;
}
```

Traversing String :- Traversing the string is one of the most important aspects in any of programming language. We may need to manipulate a very large text which can be done by traversing the text. Traversing string is sometimes different from traversing an integer array. We need to know the length of array to traverse an integer array whereas we may use the null character in case of string to identify the end the string and terminate the loop.

There are two ways to traverse a string

1. By using the length of string
2. By using the null character.

C gets() function :- The gets() function enables the user to enter some character followed by the enter key. All the characters entered by user get stored in character array. The gets() allows the user to enter the space separated strings.

Declaration :- `char[] gets(char[]);`

Reading String using gets()

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

```
Void main()
{
  char str[30];
  printf("Enter the string ");
  gets(s);
  printf("You entered %s", s);
}
```

C Puts function :- Puts() function is very much similar to printf function. The puts() function is used to print the string on the console which is previously read by using gets() or scanf() function. Puts function prints an additional newline character with string which moves the cursor to the new line on Console

Declaration :-

```
int puts(char[]);
```

Example :-

```
#include<stdio.h>
#include<string.h>
int main(){
char name[50];
printf("Enter your name:");
gets(name);
printf("Your name is :");
puts(name);
return 0;
}
```

String Manipulation functions :- In addition to input and output functions, C also has many string manipulation functions

No	Function	Description
1.	strlen(string-name)	Returns the length of string name.
2.	strcpy(destination, source)	Copies the contents of source string to destination string
3.	strcat(first_string, second_string)	concats or join first string with second string. The result of the string is stored in first string.
4.	strcmp(first_string, second_string)	Compares the first string with second string. If both strings are same it returns 0
5.	strrev(string)	returns reverse string
6.	strlwr(string)	returns string characters in lowercase
7.	strupr(string)	returns string characters in uppercase

1. strlen() :- [C string length]

The strlen() function returns the length of the given string. It does not count null character '\0'.

Syntax :- strlen(array_name);
[variable_name],

Example :-

```
#include<stdio.h>
#include<string.h>
Void main(){
char ch[20] = {'s','h','u','b','h','a','m','\0'};
    printf("length of string is %d", strlen(ch));
}
```

2.] Strcpy() :- [c copy string]

The strcpy(destination source) function copies the source string in destination

Example :-

```
#include <stdio.h>
#include <string.h>
Void main() {
char ch[20] = { 's', 'h', 'U', 'B', 'H', 'A', 'M', '\0' };
char ch2[20];
strcpy(ch2, ch);
printf("Value of second string is %s", ch2);
}
```

Syntax :-

strcpy(destination, source);

3.] Strcat() :- [c string Concalenation]

The strcat(first_string, second_string) function Concalenate two strings and result is returned to first_string.

Example :-

```
#include <stdio.h>
#include <string.h>
Void main() {
char ch[10] = { 'h', 'e', 'L', 'L', 'o', '\0' };
char ch2[10] = { 'c', '\0' };
strcat(ch, ch2);
printf("Value of first string is %s", ch);
}
```

Syntax :-

strcat(first string_second string)

4.] Strcmp() :- [c compare string]

The strcmp(first_string, second_string) function compares two string and returns 0 if both strings are equal. In below example we are using gets() function which reads string from Console

Example :-

```
#include <stdio.h>
#include<string.h>
Void main(){
char str1[20],str[20];
printf("Enter 1st string:");
gets(str1);
printf("Enter 2nd string:");
gets(str2);
if(strcmp(str1,str2)==0)
    printf("strings are equal");
else
    printf("strings are not equal");
}
```

5. strrev() :- [C Reverse string]

The strrev(string) function returns reverse of the given string :-

Example :-

```
#include <stdio.h>
#include<string.h>
Void main(){
char str[20];
printf("Enter string:");
gets(str);    // reads string from Console
printf("string is : %s",str);
printf("\n Reverse string is %s",strrev(str));
}
```

6. strlwr() :- [C string lowercase]

The strlwr(string) function returns string characters in lowercase. Syntax :- strlwr(stringname);

Example 6-

```
#include <stdio.h>
#include <string.h>
Void main() {
char str[20];
 printf("Enter string:");
 gets(str); //reads string from Console
 printf("string is %s", str);
  printf("\n lower string is %s", strlwr(str));
}
```

7.7 strupr() 6- [C string Uppercase]

The strupr(string) function returns string characters in uppercase.

Example 6-

```
#include<stdio.h>
#include<string.h>
Void main() {
char str[20];
printf("Enter string:");
gets(str);
printf("String is %s", str);
 printf("\n Upper string is %s", strupr(str));
}
```