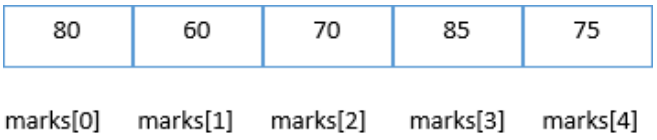
MODULE – IV

Arrays: Declaration and Initialization of one and two dimensional arrays - Multidimensional array - dynamic arrays - Character arrays and strings. Structure: Defining and processing. Structure initialization Operations on individual members Arrays of structure, Arrays within Structure, Structure and Functions- Passing to a function.

Array

Array is the group of identical elements organized in a single variable. An array is a collection of data that holds fixed number of values of same type. For example: if you want to store marks of 100 students, you can create an array for it.

int mark[100];



Initialization of Array

Arrays are of three types:

i. One-dimensional arrays

int mark[5] = {19, 10, 8, 17, 9};

Arrays have 0 as the first index not 1. In this example, mark[0]

If the size of an array is n, to access the last element, (n-1) index is used. In this example, mark[4]

ii. Two dimensional array

The 2D array is organized as matrices which can be represented as the collection of rows and columns.

The syntax to declare the 2D array is given below.

data_type array_name[rows][columns];

example

int rain[4][3];

the above example will create 4rows and 3 columns at main memory. So we can store up to 12 integers.

Row\Column	1	2	3
1			
2			
3			
4			

iii. Multidimensional arrays

In C programming, you can create an array of arrays known as multidimensional array. For example,

float x[3][4];

Here, x is a two-dimensional (2d) array. The array can hold 12 elements. You can think the array as table with 3 row and each row has 4 column.

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

Similarly, you can declare a three-dimensional (3d) array. For example,

float y[2][4][3];

Here, The array y can hold 24 elements.

Example : Each 2 elements have 4 elements, which makes 8 elements and each 8 elements can have 3 elements. Hence, the total number of elements is 24.

/* ..... Largest Number among N numbers ..... */

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int a[100],i,l,n;
  clrscr();
  printf("Enter N value ");
  scanf("%d",&n);
  for(i=1;i<=n;i++)
  {
     scanf("%d",&a[i]);
  }
  l=a[1];
  for(i=2;i<=n;i++)
  {
     if(a[i]>l)
       l=a[i];
  }
  printf("Large = %d",l);
  getch();
}
```

/* Output */

Enter Number of numbers 7
45 67 78 136 12 34 67

Large = 136

Character Array or C Strings

The string can be defined as the one-dimensional array of characters terminated by a null ('\0'). The character array or the string is used to manipulate text such as word or sentences. Each character in the array occupies one byte of memory, and the 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 the null in the memory.

There are two ways to declare a string in c language.

- By char array
- By string literal

Let's see the example of declaring string by char array in C language.

```
char ch[10]={'c', 'o', 'm', 'p', 'u', 't', 'e', 'r', "\0'};
                    or
char ch[10]="computer”
```

As we know, array index starts from 0, so it will be represented as in the figure given below.

ch

Index	0	1	2	3	4	5	6	7	8
Value	c	o	m	p	u	t	e	r	\0

Example

Let's see an example of counting the number of vowels in a string.

```
#include<stdio.h>
void main ()
{
   char s[11] = "computer";
   int i = 0;
   int count = 0;
   while(i<11)
   {
      if(s[i]=='a' || s[i] == 'e' || s[i] == 'i' || s[i] == 'u' || s[i] == 'o')
      {
         count ++;
      }
      i++;
   }
   printf("The number of vowels %d",count);
}
```

Output

The number of vowels 3

Structures

A structure is a user defined data type in C. A structure creates a data type that can be used to group items of possibly different types into a single type.

Defining a Structure

To define a structure, you must use the struct statement. The struct statement defines a new data type, with more than one member. The format of the struct statement is as follows −

```
struct [structure tag] {
   member definition;
   member definition;
   ...
   member definition;
} [one or more structure variables];
```

The structure tag is optional and each member definition is a normal variable definition, such as int i; or float f; or any other valid variable definition. At the end of the structure's definition, before the final semicolon, you can specify one or more structure variables but it is optional. Here is the way we would declare the Book structure −

```
struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
} book;
```

How to access structure elements?

Structure members are accessed using dot (.) operator.

```
#include<stdio.h>
struct Point
{
  int x, y;
};
 void main()
{
  struct Point p1 = {0, 1};
  /* Accessing members of point p1 */
  p1.x = 20;
  printf ("x = %d, y = %d", p1.x, p1.y);
}
```

Output:

x = 20, y = 1

Array of structures

Like other primitive data types, we can create an array of structures.

Example

```
struct Point
{
  int x, y;
};

void main()
{
  // Create an array of structures
  struct Point p[10];

  // Access array members
  p[0].x = 10;
  p[0].y = 20;
  p[1].x = 12;
  p[1].y = 25;

  printf("%d %d", p[0].x, p[0].y);
  printf("%d %d", p[1].x, p[1].y);
}
```

Structure and Function

A structure can be passed to any function from main function or from any sub function

PASSING STRUCTURE TO FUNCTION IN C:
It can be done in below 3 ways.

1. Passing structure to a function by value
2. Passing structure to a function by address(reference)
3. No need to pass a structure – Declare structure variable as global

In the below example student structure having id, name and percentage attributes. *record* is the variable to control entire structure.

Example:

```
#include <stdio.h>
#include <string.h>
 struct student
{
        int id;
        char name[20];
        float percentage;
};
```

```
 void func(struct student record)
{
        printf(" Id is: %d \n", record.id);
        printf(" Name is: %s \n", record.name);
        printf(" Percentage is: %f \n", record.percentage);
}

int main()
{
        struct student record;

        record.id=1;
        strcpy(record.name, "Raju");
        record.percentage = 86.5;

        func(record);
        return 0;
}
```

OUTPUT:

```
Id is: 1
Name is: Raju
Percentage is: 86.5
```