5.1. Introduction to Arrays

An array is a group of related data items that share common name. In other word an array is a data structure that store number of data items as a single entity (Object). The individual data items are called elements and all of them must have same data type.

Array is used when multiple data items that have common type and format. Therefore an array is a collection of individual data elements that is ordered (0, 1, 2, 3….), fixed size of homogeneous elements.

Suppose we have to require 10 numbers of integer type and we have to sort ascending. If we have no array we must have declared 10 integer of elements like int a1, b1, c1, c2, c3 up to 10 times of int type to store these data .When defining different variables to be stored for different numbers of times are very tedious jobs. In such case, we use array to store multiple numbers of same type to be stored at single time. The individual elements are separated by array name followed one or more subscript or index enclosed in square bracket. The individual data item can be character, integer, floating numbers. However they must be the same type and same storage class. We may used auto, register, static keyword just before deceleration variable of array type.

5.1.1. One Dimensional Array

There are several forms of an array in C, they are one dimensional and multi dimensional array. In one dimensional array, there is a single subscript or index whose value refers to individual array element which ranges from 0 to (n-1) where n is size of array e.g. int A [5] is a declaration of one dimensional array of int type. Its elements can be illustrated as:

1st element	2nd element	3rd element	4th element	5th element
A[0]	A[1]	A[2]	A[3]	A[4]
2001	2002	2003	2004	2005

The elements of an integer array A [5] are stored in continuous memory locations. It is assured that the starting memory location may be 2001. As each integer element requires 2 bytes, subsequent elements appears after gap of 2 locations. (Here int A[5] occupy 10 bytes memory area).

5.2. Declaration of Arrays

5.2.1. Declaration of 1 D Array

Syntax: storage class data type array name [size];

- Storage class refers to the storage class of array. It may be auto, static and register. It is optional.
- Data type declares of array variable hold. It may be int, float, and char. The array elements are all values of same data type. If int is used, this means that array store all data items of integer types.
- Array name is name of array. It is user defined name for array. The name of array may be any name valid for name of variable deceleration.
- Size of the array is the number of elements in the array. The size is mentioned within [] bracket. The size must be in int, constant like 10, 5. Or constant expression likes symbolic constant SIZE. #define SIZE 20, the array can be defined as int a [SIZE]. The size of an array must be specified (i.e. should not be blank) except in array initialization. Example: int num [5];

This statement tells to the compiler that num is an array of int type; num can store five integers' values. The compiler reserves 2 bytes of memory of each integer array element. Thus total memory size allocated by an integer is 10 bytes. It is individual elements are recognized by num [0], num [1], num [2], num [3], num [4]. The integer value within square bracket [] is called index of array. Index of an array always starts form 0 to size of array minus one.

Similarly, char name [10]; Here name is a character variable of size 10, it can store 10 characters.

float salary [20]; here salary is variable name having float data type array of size is 20. It can store 20 factorial values.

5.3. Initialization of Arrays

5.3.1. Initialization of 1 D array

If array elements are not given any specific values, they are supposed to contain garbage values. The values can be assigned to element at time of array declaration, which is called array initialization. Since an array his multiple elements, braces are used to denote the entire array and commas are used separate the individual values assigned to the elements in the array initialization statements.

Syntax: storage class data types array name [size] = {value1, value2, value3};

Where value1 is first elements value 2 is second element and value3 is last element. This array size is 2.

The Array Initialization May be Three Types:

1. **int a [5] = {1, 2, 3, 4, 5};**

   Here "a" is integer type array which holds 5 elements. Their values are assigned as 1,2,3,4,5(a[0]=1,a[1]=2,, a[2]=3,a[3]=4 ,a[4]=5).The array elements are stored sequentially in memory location.

2. **int b [] = {1, 2, 3};**

   Here size of array is not given; the compiler automatically sets its size accordingly to number of values present in braces. The size of array is 2 in this array initialization as three values 1,2,3 are assigned at time of initialization (a[0]=1,a[1]=2,a[2]=3).

3. **int a [10] = {1, 2, 3, 4, 5};**

   In this example the size of array has set 10 but only five elements are assigned at the time of initialization. In this situation all individual elements that are not assigned explicitly contains 0 as initialization values, as a[0]=1,a[1]=2,a[2]=3,a[3]=4,a[4]=5,a[5]=0,a[6]=0,a[7]=0,a[8]=0,a[9]=0.

5.4. Accessing Arrays Elements

The single operation that could not include entire array is not permitted in C. Thus if num and list are two similar array (data type and dimension), assignment and comparison operation of those two arrays must be carried out on the basic of element by element operation. All elements of an array can neither be set at once or one array is assigned to another.

e.g. int a [5], b [5]; then a=0; b=a; //wrong if (a<b) {…} //wrong

The particular array elements in an array are accessed by specifying the name of array, followed by sequence bracket enclosing an integer, which is called array index. The array index indicates particular elements of array which we want to access. The number of index starts form zero and ends to number one less than the size of array.

For example: float marks [5];

Then array elements are marks[0], marks[1], marks[2], marks[3] & marks[4]. We can assign float value to these individual elements directly or a loop can be used to input and output from the elements of array location.

Asking for user values in loop for each elements:

```
for(int i=0;i<5;i++)
{
        printf("Enter the value for marks[%d]:",i)
        scanf("%f", &i);
}
```

5.5. Characteristics of Array

The deceleration int a[5] is nothing but creation of 5 variables of integer type in memory . Instead of declaring five variables for five values, the programmer can define them in array of any order.

- All elements of an array share same name and they are distinguished form one another with the help of array index.
- The element number or array index in the array plays an important role for calling each element.
- Any particular element of an array can be modified separately without disturbing other elements.
- The single operations, which involve entire array, are not permitted in C. Thus if num and list are two similar array(same data type dimensions and size), then assignment operations comparison operators etc, involving these tow arrays must be carried out on comparison through element by element basic.
- Any element of an array can be assigned to another ordinary variable or array variable. int b, a[10], c=9; then b=a[2];a[5]=a[3];a[6]=c; are valid conditions.

5.6. Multidimensional Arrays

Multidimensional array are those which have more than one dimension. Multidimensional arrays are defined as same manner as one dimensional array, except that a separate pair of square brackets is required for each subscript or dimensional or index.

Thus two dimensional array will require two pairs of square brackets, similarly, the three dimensional arrays will require three pairs of square brackets and so on.

The two dimensional array is also called matrix. Multidimensional array can be defined as:

Syntax: storage class data type array name [dim1] [dim2];

Here dim1 and dim2 are positive value integer expression that indices the numbers of array elements associated with each sub-script. An m*n two dimensional array can be tabular form of value having m rows and columns. For int p [3][3] ;

	col1	**col2**	**col3**
row1	p[0][0]	p[0][1]	p[0][2]
row2	p[1][0]	p[1][1]	p[1][2]
row3	p[2][0]	p[2][1]	p[2][2]

5.6.1. Declaration of Two Dimensional Arrays

Like one dimensional array two dimensional arrays must also be declared before using it.

Syntax: storage class data type array name [row size][column size]

For int matrix [2][3]; It contains 2 rows and 3 columns it contains 6 elements.

5.6.2. Initialization of 2 D Array

Like one dimensional array multidimensional array can be initialized at the time of array defend or declaration. A few example of 2-D array initialization are:

int marks[2][3]={2,4,6},{8,10,12};

is equivalent to

- ***mark[0][0]=2***
- ***marks[0][1]= 4***
- ***mark[0][2]=6***
- ***marks[1][0]= 8***
- ***mark[1][1]=10***
- ***marks[1][2]= 12***

&

int marks[2][3]={2,4,6,8,10,12};

The first dimension may be empty like this.

int mark3[][3]= {2,4,6,8,10,12};

int p[][3]={2,4,6,8,10};

p[0][0]=2,p[0][1]=4,p[0][2]=6, p[1][0]=8,p[1][1]=10,p[1][2]=0

Note: if value provided while initializing are less than an array, zero is assigned to remaining elements.

All above example are valid while initializing array, it is necessary to mention second dimensional column where as first dimension row is optional. This following are invalid initialization:

int marks[3][]={2,4,6,10,12};//wrong

int marks3[][]={2,4,6,8,10};//error

5.6.3. Accessing 2d Array Elements

In 2D array first dimension specifies number of rows and second specifies columns. Each row contains elements of many columns. 2D array contains multiple rows.

Thus 2D array is an array of 1D array. As each size row will contain elements of many columns, the columns index ranges from 0 to size-1 inside every row in one dimensional representation. So columns index changes faster than row index as one dimensional representation of array inside computer is traversed.

2D array is traversed row by row (every columns elements in first row are traversed first and then columns elements of second row are traversed and so on. The nested loop is used to traverse the 2D array.

Let's consider following 2D array marks, size of 4*3 matrix having 4 row and 3 columns.

35 10 11

34 90 76

13 8 5

76 4 1

i.e.: 35 10 11 34 90 76 13 8 5 76 4 1

To access a particular elements of 2D array we have to specify array name followed by two square brackets with row and columns number inside it.

For example: marks[0][0]=35, marks[0][1]=10

5.7. Strings

Strings are array of characters. They are arranged one after another in memory. Thus character array is also known as string.

Each character within string will be stored within each element of the array. Strings are used by "%c" to manipulate text such as word and sentences. A string is always terminated by a null character "\0". The terminating null character is important because it is only way string handling functions can known where string ends. Normally each character is stored in one byte, and successive characters of the string are stored in successive bytes.

Initialization of Strings:

char name[]={'R','A','M'};

Then string name is initialized to "RAM". This technique is also valid but offers special way to initialize the sting as:

char name[]={"RAM"};

The characters of string are enclosed within the double quotes.

Arrays of Strings:

String is array of characters. Thus an array of string is 2D array of characters. This is explained with the help of following example:

/* program to read name of 5 different person using array of strings & display them. */

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char name[5][10]; int i;
        printf("enter name of 5 students");
        for(i=0;i<5;i++)
                scanf("%s",name[i]);
        printf("The enter name are");
        for(i=0;i<5;i++)
                printf("\nThe%d name is%s",i,name[i] );
        getch();
}
```

5.7. String Handling Functions

The library built in functions strlen (), strcpy (), strcat (), strcmp () strrev () etc are used for string manipulation in C. These functions are defined in header file string.h

5.7.1. strlen ()

The strlen () function returns integer which denotes length of string passed. The length of string is number of characters presented in it, excluding terminating null character.

Syntax: integer variable=strlen (string);

WAP to find the length of Input from user using function.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char name[10];
        int len;
        clrscr();
        printf("\n Enter Your Name");
        gets(name);
        len=strlen(name);
        printf("\n the number of character in our yname is%d",len);
        getch();
}
```

5.7.2. strcpy()

The strcpy () function copies one string to another. The function accepts two strings as parameters. The first string copies the string of second string character by character into first one up to and including the null character of second string.

Syntax: strcpy (destination string, sources string);

WAP to copy one string to another

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char name[]="Hello World";
        char s[25];
        clrscr();
        strcpy(s,name) ;
        printf("\n the Value s=%s\t name =%s",s,name);
        getch();
}
```

5.7.3. strcat ()

This function concatenates two string i.e it appends one string at the end of another. This function accepts two strings as parameter and stores the contents of the second string at end of the first string.

Syntax: strcat (string1, string2);

WAP to concatenate two strings

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char fname[20]="Hello ", lname[10]="World";
        clrscr();
        strcat(fname,lname) ;
        printf("\n full name is%s",fname);
        getch();
}
```

5.7.4. strcmp()

This function compares two strings to find out whether they are same or different. This function is useful for searching string as arranged in a dictionary order. This function accepts two strings as parameters and returns an integer whose value is ,

- Less than 0 if first string is less than second string
- Equal to 0 if both are same
- Greater than 0 if the first string is greater than second

Two strings are compared character by character until there is a mismatch or end of strings. When two characters in two strings differ, the string which has the character with a higher ASCII values is displayed.

Syntax: strcmp(string1, string2);

WAP to illustrate to use of strcmp () function

```
#include<stdio.h>
#include<string.h>
#include <conio.h>
void main()
{
        char fname[20],lname[10];
        int diff;
        printf("enter first name"); gets(fname);
        printf("Enter second naame"); gets(lname);
        diff=strcmp(fname,lname);
        if(diff<0)
                printf("%s is greater than %s by value of %d",fname,lname,diff);
        else if(diff<0)
                printf("%s is less than %s by value of %d",lname,fname,diff);
        else
                printf("%s is same length %s ",fname,lname);
        getch();
}
```

5.7.5. strrev()

This function is used to reverse all characters in string except null character at the end of string. The reverse of string “abc” is “cba”.

Syntax: strrev(str1);

WAP to illustrate strrev function

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char name1[15]="Hello World", name2[15];
        strcpy(name2,name1);
        strrev(name1);
        printf("The reversed string of original string %s is%s",name2,name1);
        getch();
}
```