Unit
4

Arrays and Pointers

4.1 INTRODUCTION TO AN ARRAY

An array is a fixed-size sequenced collection of elements of the same data type. An array is a variable that can store multiple values. For example, if you want to store 100 integers, you can create an array for it

```
int data[100];
```

4.2 CHARACTERISTIC OF AN ARRAY

- An array holds elements that have the same data types.
- Array elements are stored in subsequent memory locations.
- Array name represents the address of the first elements.
- An array index by default starts from 0 and ends with the index no(n-1).
- The size of an array cannot change at run time.

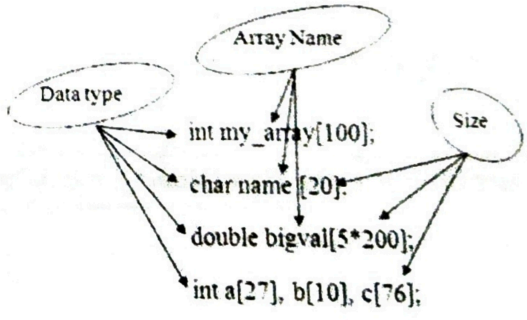
4.3 ONE DIMENSIONAL ARRAY

An array can be used to represent a list of numbers, or a list of names. Types of arrays:

1. One-dimensional arrays
2. Two-dimensional arrays
3. Multidimensional arrays

A list of items can be given one variable name using only one subscript and such a variable is called a one-dimensional array.

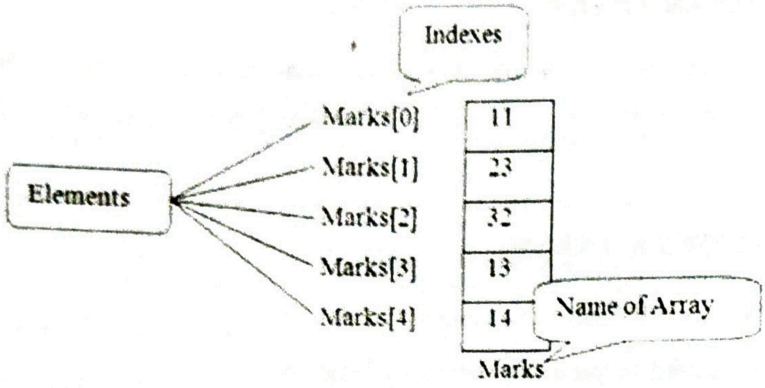
4.3.1 Declaring One Dimensional Array

Syntax:	Example:
data_type array_name[size];	

Example of declaring one dimensional array:

```
int Marks[5];
```

The above statement declares an array 'Marks', which reserves 5 memory locations for storing integer type of data. The names of the memory locations are given as Marks[0], Marks[1], Marks[2], Marks[3] and Marks[4]. The number within the square brackets is known as index or subscript.



Here above statement declare array of size 5 then it occupies 5 memory locations and it can store maximum 5 integer elements that mean we can store 5 different integer values inside it. The subscript (index) or array is start with the 0 and up to 4 (size –1).

4.3.2 Initializing One Dimensional Array

There are two ways to initialize one dimensional array:

Compile time initialization : The general form of array initialization is

type arrayname[size] = { list of values };

Example:

```
int a[3]= {1,2,3}
char name[]={'h','e','l','l','o','\0'};
```

When the size of an array is omitted, that time the compiler allocates enough space for all initialized elements.

Run time initialization :

```
for(i=0;i<10;i++)
{
scanf("%d",&sum[i]);
}
```

Representation of a one-dimensional array in memory

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

Then

a[0] = 1

a[1] = 2

a[2] = 3

a[3] = 4

a[4] = 5

4.3.3 Accessing One Dimensional Array Elements

You can access elements of an array by indices. Suppose you declared an array a[5] as above. The first element is a[0], the second element is a[1] and so on.

Example:

```
printf("%d", a[2]);
```

the above statement will print 3

Program example to access array elements.

```
/* ----------PROGRAM TO ACCESSING ARRAY ----------*/
#include<stdio.h>
#include<conio.h>
void main()
{
int score[10];                  /* Declaring an array of  type int with 10 elements */
int i;                          /* Index for the array */
for (i=0; i<10; i++)
{
printf("Enter Marks: ");
scanf("%d",& Marks[i]);
}
getch();
}
```

Output:

```
Enter Marks:
10
22
34
23
54
12
10
56
24
56
```

4.4 TWO DIMENSIONAL ARRAY

The two-dimensional array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns.

4.4.1 Declaring Two-dimensional Array

Two dimensional arrays are declared as follows:

```
type  arrayname [row_size] [column_size];
```

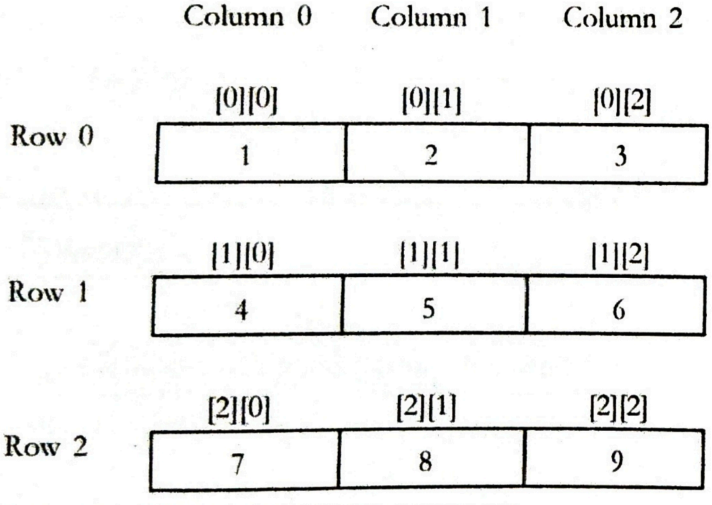
Example:

```
int table[2][3]={0,0,0,1,1,1};
        OR
for(i=0;i<2;i++)
{
for(j=0;j<3;j++)
{
scanf("%d',&table[i][j]);
}
}
```

Representation of a two-dimensional array in memory

If int a[3][3] = {1,2,3,4,5,6,7,8,9};

Then



4.4.2 Initialization of Two-dimensional Array

There are two ways to initialize a two-Dimensional array during declaration.

```
int arr[2][4] = {
    {10, 11, 12, 13},
    {14, 15, 16, 17}
};
```

OR

```
int arr[2][4] = { 10, 11, 12, 13, 14, 15, 16, 17};
```

The two-dimensional array can be declared and defined in the following way.

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

4.4.3 Accessing Two-dimensional Array Elements

We will learn how to access two-dimensional array elements using an example of matrix.

Example: Storing elements in a matrix and printing it.

```
#include <stdio.h>
void main ()
{
  int arr[3][3],i,j;
  for (i=0;i<3;i++)
  {
    for (j=0;j<3;j++)
    {
      printf("Enter a[%d][%d]: ",i,j);
      scanf("%d",&arr[i][j]);
    }
  }
  printf("\n printing the elements ....\n");
  for(i=0;i<3;i++)
  {
    printf("\n");
    for (j=0;j<3;j++)
    {
      printf("%d\t",arr[i][j]);
    }
  }
}
```

Output

```
Enter a[0][0]: 56
Enter a[0][1]: 10
Enter a[0][2]: 30
Enter a[1][0]: 34
Enter a[1][1]: 21
Enter a[1][2]: 34
```

```
Enter a[2][0]: 45
Enter a[2][1]: 56
Enter a[2][2]: 78
```

```
printing the elements ....

56      10      30
34      21      34
45      56      78
```

4.4.4 Multi-Dimensional Arrays

The general form of a multi-dimensional array is

```
type  arrayname [s1][s2][s3].....[sn]
```

Example:

```
int survey[3][5][12];
float table[5][4][5][3];
```

4.5 INTRODUCTION TO STRINGS

A string is a sequence of characters that is treated as a single data item. Any group of characters defined between double quotation marks (" ") is a string constant. Each string in C is ended with the special character called NULL character ('\0').

4.5.1 Declaration of Strings

The general form of declaration of a string variable is

```
char string_name [size];
```

Example:

```
char city[9]="new york";
```

OR

```
char address[10];
scanf("%s",address);
```

OR

```
gets(address);
```

4.5.2 gets() and puts() functions

gets() function

The gets() function is similar to scanf() function . The gets() function enables the user to enter some characters followed by the enter key. All the characters entered by the user get stored in a character array. The null character is added to the array to make it a string. The gets() allows the user to enter the space-separated strings. It returns the string entered by the user.

Declaration

```
char[ ] gets(char[ ]);
```

puts() function

The **puts()** function is similar to printf() function.

Declaration

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

Example using gets() and puts() functions

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

Output:

```
Enter your name: Jainik
Your name is: Jainik
```

4.5.3 String-Handling Functions

The following table provides most commonly used string handling function and their use...

Function	Syntax (or) Example	Description
strcpy()	strcpy(string1, string2)	Copies string2 value into string1
strncpy()	strncpy(string1, string2, 5)	Copies first 5 characters string2 into string1
strlen()	strlen(string1)	returns total number of characters in string1
strcat()	strcat(string1,string2)	Appends string2 to string1

Function	Syntax (or) Example	Description
strncat()	strncpy(string1, string2, 4)	Appends first 4 characters of string2 to string1
strcmp()	strcmp(string1, string2)	Returns 0 if string1 and string2 are the same;less than 0 if string1<string2; greater than 0 if string1>string2
strncmp()	strncmp(string1, string2, 4)	Compares first 4 characters of both string1 and string2
strcmpi()	strcmpi(string1.string2)	Compares two strings, string1 and string2 by ignoring case (upper or lower)
stricmp()	stricmp(string1, string2)	Compares two strings, string1 and string2 by ignoring case (similar to strcmpi())
strlwr()	strlwr(string1)	Converts all the characters of string1 to lower case.
strupr()	strupr(string1)	Converts all the characters of string1 to upper case.
strdup()	string1 = strdup(string2)	Duplicated value of string2 is assigned to string1
strchr()	strchr(string1, 'b')	Returns a pointer to the first occurrence of character 'b' in string1
strrchr()	'strrchr(string1, 'b')	Returns a pointer to the last occurrence of character 'b' in string1
strstr()	strstr(string1, string2)	Returns a pointer to the first occurrence of string2 in string1
strset()	strset(string1, 'B')	Sets all the characters of string1 to given character 'B'.
strnset()	strnset(string1, 'B', 5)	Sets first 5 characters of string1 to given character 'B'.
strrev()	strrev(string1)	It reverses the value of string1

1. strlen()

Thestrlen() function counts and returns the number of characters in a string. It takes the form

```
n = strlen(s1);
```

Example : If s1="college",

n = strlen(s1);

Output will be n=7.

2. strcpy()

The strcpy() function copy one string into another string. It takes the form

```
strcpy(s1,s2);
```

Example : If s1="the" and s2="their",

strcpy(s1,s2);

output will be s1="their" and s2="their". Because it will copy string s2 to s1.

3. **strcat()**

The strcat() function joins two strings together. It takes the following form:

```
strcat(s1,s2);
```

Example : strcat("My ","Name");

Output will be 'My Name'.

4. **strcmp()**

The strcmp() function compares two strings. If both are equal, it returns 0. If first string is alphabetically y greater than the second string,it returns positive number, otherwise returns negative number. It takes the form:

```
strcmp(s1,s2);
```

Example : If s1="there" and s2="their",

strcmp(s1,s2);

Output will be 9. Because it will compare characters and here, difference between ASCII 'r' and 'i' is 9.

5. **strrev()**

The strrev() function reverses the string and original string is overwritten.It takes the form:

```
strrev(s1);
```

Example : If s1="there",

strrev(s1);

Output will be s1="ereht".

6. **strncpy()**

The strncpy() function copies only the left-most n characters of the second string to the first string. It takes the form:

```
strncpy(s1,s2,n);
```

Example : If s1="there", s2="when" and n=2,

strncpy(s1,s2,n);

Output will be 'where'.

7. **strncat()**

The strncat() function concatenate the left-most n characters of second string to the end of first string. It takes the form:

```
strncat(s1,s2,n);
```

Example : If s1="there", s2="where" and n=3,

strncat(s1,s2,n);

Output will be 'therewhe'.

8. strcmpi()

The strcmpi()function compares two strings ignoring the case. If both are equal, it returns 0. If first string is alphabetically y greater than the second string,it returns positive number, otherwise returns negative number. It takes the form:

```
strcmpi(s1,s2);
```

Example : If s1="ThEre" and s2="there",

strcmpi(s1,s2);

Output will be 0. Because both strings are equal (ignoring the case).

9. strncmp()

The strncmp() function compares the left-most n characters of one string to another string. It takes general form:

```
strncmp(s1,s2,n);
```

Example : If s1="there", s2="their" and n=3,

strncmp(s1,s2,n);

Output will be 0.Because in both stringsleft-most 3 charactersare equal.

10. strncmpi()

The strncmpi() function compares the left-most n characters of one string to another string ignoring the case. It takes general form:

```
strncmpi(s1,s2,n);
```

Example : If s1="tHEre", s2="their" and n=3,

strncmpi(s1,s2,n);

Output will be 0.Because in both stringsleft-most 3 charactersare equal (ignoring the case).

11. strupr()

The strupr() function converts the string to uppercase.

Example : If s1="tHEre",

strupr(s1);

Output will be s1="THERE".

12. strlwr()

The strlwr() function converts the string to lowercase.

Example : If s1="tHEre",

strlwr(s1);

Output will be s1="there".

13. strstr()

It is a two parameter function that can be used to locate a sub string in a string. This takes the forms: