UNIT III

Arrays Introduction

The array means arranged things. It is a collection of similar type of elements that have sequential memory. The term "data structure" is arranging data values in a specific order. There are different types of data structures and array is linear data structure.

When a program needs to store large number of data elements of similar type, then instead of using individual elements, the arrays can be preferred, and with arrays, data manipulation becomes easy.

Declaration and initialisation of Arrays

In C-language, the [ ] symbol is used while declaring and manipulating the arrays. The [ ] are used to specify no. of elements to be grouped while declaring.

Syntax : data_type array_name[int size];

Ex: int a[5];

When an array is declared, the C-RE gives index numbers to each element for identification as all elements are identified with same name. The index number of array elements starts with ZERO. So the index of last element is always size-1.

The same [ ] are used to specify the index number of element while manipulating the array. i.e. the array members can be accessed with their index numbers.

Storing Values in Array & Accessing elements of the Array –

Arrays can be stored with values either by initializing or by taking input. The initializing can be done in two ways.

int arr[5] = { 10, 20, 30,40, 50}; in this case, the number of initialisers must be either matching with size of array or less than array. If less initialisers given when size is specified, then the other elements are initialized with ZEROs.

or

int arr[ ]={10,20,30,40,50}; in this case, the size of array is based on number of initializers.

The loops can be used to read data from keyboard and store into the array.

Ex:

```
for(i=0;i<5;i++)
    scanf("%d",&arr[i]);
```

we can access array elements by referring the index number as subscript within [ ] with array name.

Ex:

```
for(i=0;i<5;i++)
    printf("%d",&arr[i]);
```

Operations on Arrays :

There are different operations that can be performed with arrays like,

traversal : moving across the array elements using a loop.

Copying: copying elements from an array or into an array

Reversing : reversing array elements

sorting: arranging array elements either in ascending or descending order.

inserting / deleting : inserting or deleting values from array

searching : searching for given value

merging : merging multiple arrays into one array.

Q. Explain about arrays in C
Q. write how to store and access values in an array
Q. explain arrays with different operations that can be done on arrays.

2D and Multi dimensional arrays: the arrays can be used to manage large data by providing multiple index references, for easy storage and manipulation. Irrespective of number of dimensions specified, all the array elements will have sequential memory allocated.

In a 2-dimensional array the size of each dimension is specified with in a separate [ ].

Ex : int arr[2][3];

In the above 2-D array, the memory allocated is sequential as given below:

arr

[0][0] [0][1] [0][2] [1][0] [1][1] [1][2]

The above 2-D array can be manipulated using two loops, one to refer first index number and the second referring second index number.

We can initialize a 2D array as :

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

we can use two loops each referring an index.

```
for( i=0;i<2;i++)
for(j=0;j<3;j++)
    printf("%d ",arr[i][j]);
```

A 2-D array can be logically felt as rows and columns by displaying content row wise / column wise with below code.

```
for( i=0;i<2;i++)
{
    for(j=0;j<3;j++)
        printf("%d ",arr[i][j]);
    printf("\n");
}
```

It can be logically assumed in a tabular format as given below :

[0][0]	[0][1]	[0][2]
[1][0]	[1][1]	[1][2]

arr

Similarly, the arrays can have any number dimensions, and the size of each dimension is specified with in a separate [ ].

Mathematically, the two dimensional arrays are treated as matrices and this manipulation of matrices is much important in machine learning.

Ex: int arr[2][3][2];

Irrespective of number of dimensions, the array elements will have sequential memory. But in practical live programming a 2-D array is much used as it can be used to store large data and managed in a tabular format. The multi-dimensional arrays that have more than 2 dimensions are not much used in regular programming and it is difficult to represent them.

Q. Explain 2-dimensional and multi dimensional arrays?

Strings

String is sequence of characters. In C, the strings are managed with an array of char data type.
Ex: char name[20];

We can initialize the string as char name[]= "Gnanambica";

When a string is assigned to a char array while initializing, there will be null value represented as '\0' stored at end of string for identification.

The C has a special format specifier %s to be used while reading or writing strings. The %s won't read spaces in a string. We can also specify the characters to be read as format specifier instead of %s like "%[a-z]". The ^ (caret) can be used to specify the character NOT to be read. Many times programmers use "%[^\n']" as format specifier to read all symbols spaces etc till user hits ENTER.

For easy manipulation of strings, the string.h library has number of pre-defined functions like:

strcpy(char dest[ ], char source[ ]) ; copies source [ ] into dest[].
strcat(char dest[ ], char source[ ]) ; concatenates source [ ] to dest[].
strlen(char[ ]) : returns length of string
strcmp(char ch1[ ], char ch2 [ ]) : compares two strings lexicographically with their ASCII values.
strrev(char ch[ ]) : reverses given string.

Though the group of characters is a string, we may get a requirement of handling each character individually while programming. For character handling, the ctype.h library has number of pre-defined functions like:

islower(ch) : returns true if lowercase letter and false if not
isupper(ch) : returns true if upper case letter

tolower(ch) : converts into lowercase
toupper(ch) : converts into uppercase
isspace(ch) : returns true if it is a space
isalpha(ch) : returns true if it is alphabet
isdigit(ch) : returns if it is a digit
isalnum(ch) : returns true if it is either alphabet or a number

Q. Explain string and character manipulation in C.
