GXEST204

PROGRAMMING IN C

MODULE 2

CO2 : Develop C programs using arrays, matrices, and strings.

Arrays - Single dimensional arrays, Defining an array, Array initialization, Accessing array elements; Enumerated data type; Type Definition; Two-dimensional arrays – Defining a two-dimensional array; Programs for matrix processing; Programs for sequential search; Bubble sort; Strings – Declaring a string variable, Reading and displaying strings, String related library functions – Programs for string matching.

INTRODUCTION TO ARRAYS

- An array is defined as the collection of similar type of data items stored at contiguous memory locations.
- Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc.
- It also has the capability to store the collection of derived data types, such as pointers, structure, etc.
- The array is the simplest data structure where each data element can be randomly accessed by using its index number
- C array is beneficial if you have to store similar elements.
- For example, if we want to store the marks of a student in 6 subjects, then we don't need to define different variables for the marks in the different subject.
- Instead of that, we can define an array which can store the marks in each subject at the contiguous memory locations.
- By using the array, we can access the elements easily.
- Only a few lines of code are required to access the elements of the array.

Declaration of C Array

We can declare an array in the c language in the following way.

data_type array_name[array_size];

Now, let us see the example to declare the array

int marks[5];

Here, int is the data_type, marks are the array_name, and 5 is the array_size.

Initialization of C Array

The simplest way to initialize an array is by using the index of each element.

We can initialize each element of the array by using the index.

Consider the following example.

marks[0]=80; marks[1]=60; marks[2]=70; marks[3]=85; marks[4]=75;

There are various ways in which we can declare an array.

It can be done by specifying its type and size, by initializing it or both.

1. **Array declaration by specifying size**

```
// Array declaration by specifying size int arr1[10];
// With recent C/C++ versions, we can also // declare an array of user specified size
int n = 10;
int arr2[n];
```

2. **Array declaration by initializing elements**

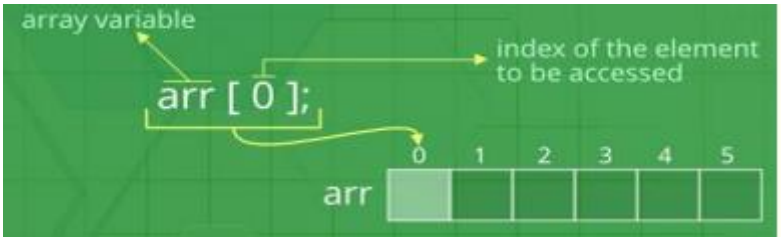
```
// Array declaration by initializing elements
int arr[] = { 10, 20, 30, 40 };
// Compiler creates an array of size 4.
// above is same as
int arr[4] = {10, 20, 30, 40};
```

3. **Array declaration by specifying size and initializing elements**

```
// Array declaration by specifying size and initializing
// elements
int arr[6] = { 10, 20, 30, 40 } ;
// Compiler creates an array of size 6, initializes first
// 4 elements as specified by user and rest two elements as 0.
// above is same as
int arr[] = {10, 20, 30, 40, 0, 0};
```

Accessing Array Elements

- Array elements are accessed by using an integer index.
- Array index starts with 0 and goes till size of array minus 1.



Program to find sum of n numbers

```
#include <stdio.h>
int main()
{
        int n, sum = 0, c, value;
         printf("How many numbers you want to add?\n");
        scanf("%d", &n);
        printf("Enter %d integers\n", n);
        for (c = 1; c <= n; c++)
        {
                scanf("%d", &value);
                sum = sum + value;
         }
        printf("Sum of the integers = %d\n", sum);
        return 0;
}
```

Enumerated data type

The enumerated data type is also known as the enum. Now, enum is not a basic, but a user-defined form of data type that consists of various integer values, and it functions to provide these values with meaningful names. Using the enum data type in C language makes any program much easier to maintain as well as understand. We define enums using the keyword "enum".

SYNTAX:- enum flag{integer_constant1, integer_constant2,…..integter_constantN};

In the above declaration, we define the enum named as flag containing 'N' integer constants. The default value of integer_const1 is 0, integer_const2 is 1, and so on. We can also change the default value of the integer constants at the time of the declaration.

For example: enum fruits{mango, apple, strawberry, papaya};
The default value of mango is 0, apple is 1, strawberry is 2, and papaya is 3. If we want to change these default values, then we can do as given below:

```
enum fruits
{
    mango=2,
```

```
    apple=1,
    strawberry=5,
    papaya=7,
};
```

Let's create a simple program of enum.

```
#include <stdio.h>
enum weekdays{Sunday=1, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday};
int main()
{
        enum weekdays w; // variable declaration of weekdays type
        w=Monday; // assigning value of Monday to w.
        printf("The value of w is %d",w);
        return 0;
}
```

Output

The value of w is 2

TYPE DEFINITION IN C

The **typedef** is a keyword used in C programming to provide some meaningful names to the already existing variable in the C program. It behaves similarly as we define the alias for the commands. In short, we can say that this keyword is used to redefine the name of an already existing variable.

Syntax: typedef <existing_name> <alias_name>

In the above syntax, '**existing_name**' is the name of an already existing variable while '**alias name**' is another name given to the existing variable.

For example, suppose we want to create a variable of type **unsigned int**, then it becomes a tedious task if we want to declare multiple variables of this type. To overcome the problem, we use **a typedef** keyword.

Eg**:- typedef unsigned int unit**;

MULTI DIMENSIONAL ARRAYS

Multidimensional arrays are often known as array of the arrays

Multi Dimensional Array:

1. Array having more than one subscript variable is called Multi-Dimensional array.

2. Multi Dimensional Array is also called as Matrix.

Syntax: DATATYPE ARRAYNAME [row subscript][column subscript];

Example: int name[50][20];

Initialization:

int a[3][3] = { 1, 2, 3 5, 6, 7 8, 9, 0};

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

In the above example we are declaring 2D array which has 2 dimensions.

First dimension will refer the row and 2nd dimension will refer the column.

Two Dimensional Array requires

- Two Subscript Variables
- Two Dimensional Array stores the values in the form of matrix.
- One Subscript Variable denotes the "Row" of a matrix.
- Another Subscript Variable denotes the "Column" of a matrix

2-D arrays are stored in contiguous memory location row wise.

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Program to add two matrices

```
#include <stdio.h>
int main()
{
        int m, n, c, d, first[10][10], second[10][10], sum[10][10];
        printf("Enter the number of rows and columns of matrix\n");
        scanf("%d%d", &m, &n);
        printf("Enter the elements of first matrix\n");
        for (c = 0; c < m; c++)
        {
                for (d = 0; d < n; d++)
                {
                        scanf("%d", &first[c][d]);
                }
```

```
        }
        printf("Enter the elements of second matrix\n");
        for (c = 0; c < m; c++)
        {
                for (d = 0 ; d < n; d++)
                {
                        scanf("%d", &second[c][d]);
                }
        }
        printf("Sum of entered matrices:-\n");
        for (c = 0; c < m; c++)
        {
                for (d = 0 ; d < n; d++)
                {
                        sum[c][d] = first[c][d] + second[c][d];
                        printf("%d\t", sum[c][d]);
                }
                printf("\n");
        }
        return 0;
}
```

STRINGS IN C

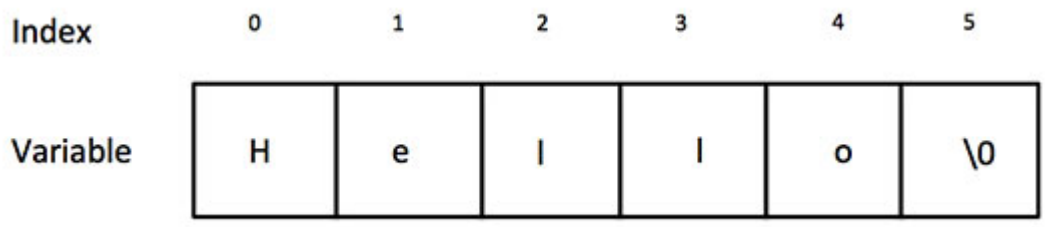
A **string in C** is a one-dimensional array of char type, with the last character in the array being a "null character" represented by '\0'.

Initializing String Without Specifying Size

C lets you initialize an array without declaring the size, in which case the compiler automatically determines the array size.

char greetings[]={'H','e','l','l','o','\0'};

The array created in the memory can be schematically shown as follows −



The length of the string doesn't include the null character.

How to read a string and display

- For character we need to use scanf("%c", &character);
- For string we need to use scanf("%s", string);
- And for string with spaces we need to use fgets() function. Here the first parameter is string, second is the size and last one is **stdin** to take input from console.

Example:-

```
#include <stdio.h>
int main() {
  char str[20];
  printf("Enter a string:");
  scanf("%s",str);
  printf("Entered string is %s", str);
}
```

Output

Enter a string:Welcome

Entered string is Welcome

Loop Through a String

You can loop through a string (character array) to access and manipulate each character of the string using the for loop or any other loop statements.

```
#include<stdio.h>
#include<string.h>
int main()
{
        char greetings[]={'H','e','l','l','o','\0'};
        for(int i=0;i<5;i++)
        {
                printf("%c",greetings[i]);
        }
        return 0;
}
```

C program to find length of string without using string function

```
#include <stdio.h>
int main() {
   char str[20];
   int i,len=0;
   printf("Enter a string:");
   scanf("%s",str);

   for (i = 0; str[i] != '\0'; ++i)
   {
        len=len+1;
   }
   printf("Length of the string: %d", len);
```

```
    return 0;
}
```

Output

```
Enter a string:Welcome
Length of the string: 7
```

C program to check whether the string is palindrome or not without using string functions

```
#include <stdio.h>
int main() {
  char str[20];
  int i,len=0,flag=0;
  printf("Enter a string:");
  scanf("%s",str);
  for(i=0;str[i]!='\0';i++)
  {
     len=len+1;
  }
  for(i=0;i<len/2;i++)
  {
     if(str[i]!=str[len-i-1])
     {
        flag=1;
        break;
     }
  }
  if(flag==1)
  {
     printf("String is not palindrome");
  }
  else
  {
     printf("String is palindrome");
  }
}
```

Output

```
Enter a string: malayalam
String is palindrome
```

STRING FUNCTIONS

C supports a large number of string handling functions in the standard library "string.h"
String handling functions are defined under "string.h" header file.

Function	Work of Function
strlen()	computes string's length
strcpy()	copies a string to another
strcat()	concatenates(joins) two strings
strcmp()	compares two strings
strlwr()	converts string to lowercase
strupr()	converts string to uppercase

strlen( ) Function:

strlen( ) function is used to find the length of a character string.

Ex:

int n;

char st[20] = “Bangalore”;

n = strlen(st);

This will return the length of the string 9 which is assigned to an integer variable n.

Note that the null charcter „\0“ available at the end of a string is not counted.

strcpy( ) Function:

strcpy( ) function is used to copy from one string to another string.

Ex :

char city[15];

strcpy(city, “BANGALORE”) ;

This will assign the string “BANGALORE” to the character variable city.

* Note that character value like city = “BANGALORE”; cannot be assigned in C language.

strcat( ) Function:

strcat( ) function is used to join character, Strings.

When two character strings are joined, it is referred as concatenation of strings.

Ex:

char city[20] = “BANGALORE”;

char pin[8] = "-560001";

strcat(city , pin);

This will join the two strings and store the result in city as "BANGALORE – 560001".

Note that the resulting string is always stored in the left side string variable.

strcmp( ) Function:

strcmp ( ) function is used to compare two character strings.

It returns a 0 when two strings are identical. Otherwise it returns a numerical value which is the different in ASCII values of the first mismatching character of the strings being compared.

char city[20] = "Madras";

char town[20] = "Mangalore";

strcmp(city, town);

This will return an integer value – 10 which is the difference in the ASCII values of the first mismatching letters d and n. Note that the integer value obtained as the difference may be assigned to an integer variable as follows:

int n;

n = strcmp(city, town);

strchr()

strchr() is a predefined function in c. It is used to find the first occurrence of a character in a string. It checks the whether the given string is present in the given string or not. If a character is found it returns the pointer to it otherwise it returns a null pointer.

Eg:- s[20]="Good morning";

strchr(s,o); returns 1 as position of index of first occurrence of o;

strlwr()

strlwr() returns is used to convert the given string to lower case.

char str[]="GOOD MORNING"

strlwr(str) – returns the strings as good morning

strupr()

the strupr() function is used to converts a given string to uppercase.

char str[ ] = "good morning";

strupr(str)- returns the string as GOOD MORNING

SEQUENTIAL SEARCHING

Linear search is a type of sequential searching algorithm. In this method, every element within the input array is traversed and compared with the key element to be found. If a match

is found in the array the search is said to be successful; if there is no match found the search is said to be unsuccessful and gives the worst-case time complexity.

The algorithm for linear search is relatively simple. The procedure starts at the very first index of the input array to be searched.

Step 1 − Start from the 0th index of the input array, compare the key value with the value present in the 0th index.

Step 2 − If the value matches with the key, return the position at which the value was found.

Step 3 − If the value does not match with the key, compare the next element in the array.

Step 4 − Repeat Step 3 until there is a match found. Return the position at which the match was found.

Step 5 − If it is an unsuccessful search, print that the element is not present in the array and exit the program.

Program- Linear searching

```
#include<stdio.h>
void main()
{
   int num,arr[20];
   int i,key,element_found = 0;
   printf("Enter number of elements you would like to take as input: ");
   scanf("%d",&num);
   printf("\nEnter all the elements of your choice:");
   for(i=0;i<num;i++)
   {
      scanf("%d",&arr[i]);
   }
   printf("\nEnter the key element that you would like to be searched: ");
   scanf("%d",&key);
  for(i=0;i<num;i++)
   {
      if(key==arr[i] )
      {
         element_found=1;
         break;
      }
```

```
    }
    if(element_found==1)
        printf("we got the element at index %d",i+1);
    else
        printf("we haven not got element at any index in the array\n");
}
```

BUBBLE SORT

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity are of $O(n^2)$ where **n** is the number of items.

We assume **list** is an array of **n** elements. We further assume that **swap** function swaps the values of the given array elements.

Step 1 − Check if the first element in the input array is greater than the next element in the array.

Step 2 − If it is greater, swap the two elements; otherwise move the pointer forward in the array.

Step 3 − Repeat Step 2 until we reach the end of the array.

Step 4 − Check if the elements are sorted; if not, repeat the same process (Step 1 to Step 3) from the last element of the array to the first.

Step 5 − The final output achieved is the sorted array.

Program

```
#include <stdio.h>
int main()
{
    int  n,j,i,swap,array[25];
    printf("Enter number of elements\n");
    scanf("%d",&n);
    printf("Enter %d integers\n",n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&array[i]);
    }
```

```
    for(i=0;i<n-1;i++)
    {
        for(j=0;j<n-i-1;j++)
        {
            if(array[j]>array[j+1])
            {
                swap=array[j];
                array[j]=array[j+1];
                array[j+1]=swap;
            }
        }
    }
    printf("Sorted list in ascending order:\n");
    for(i=0;i<n;i++)
        printf("%d\n", array[i]);
    return 0;
}
```