CHAPTER – 5
Array and Strings

Introduction to Array

An array is the collective name given to a group of similar quantities. Or an array is a set of similar data elements which are stored in consecutive memory locations under a common variable name. The individual values in an array are called elements. Array is sets of values of the same type, which have single name followed by an index. The similar element of an array would be all integers or all floats or all characters. Usually, the array of character is called string.

The memory space taken by various array elements are shown below:

int → 2 byte memory space
float → 4 byte memory space
char → 1 byte memory space

Example: int X[5]= {3, 10, 15,8, 7};

Array index	X[0]	X[1]	X[2]	X[3]	X[4]
Array value	3	10	15	8	7
Memory address	20030	20032	20034	20036	20038

Example: float X[5]= {3.2, 10.8, 8.2, 2.4, 3.3};

Array index	X[0]	X[1]	X[2]	X[3]	X[4]
Array value	3.2	10.8	8.2	2.4	3.3
Memory address	20030	20034	20038	20042	20046

Example: char X[5]= {N, E, P, A, L};

Array index	X[0]	X[1]	X[2]	X[3]	X[4]
Array value	N	E	P	A	L
Memory address	20030	20031	20032	20033	20034

Q. Why array is useful?

It is an inconvenient method to declare unique variable names for all the similar data items. To solve this problem a new concept is developed known as array. An array is a convenient method which is used to represent collections of similar data elements by a single unique name.

Limitations of array:

Conventional arrays are static in nature i.e. memory is allocated in the beginning of the program and it cannot be changed dynamically. If n is the size of an array and only m elements are needed then n - m locations are unnecessarily wasted.

Array Declaration

Declaring the name and type of an array and setting the number of elements in the array is known as dimensioning(declaring) the array. It must be declared before it is used like other variables. In the array declaration, we must define the type of the array, name of the array and number of subscripts in the array. In general, one-dimensional array may be expressed as:

data_type array_name[size];

Example:

int marks[300];

where, int = Data type or Type of variable
marks = Name of array variable
300 = Dimension (size) of array
[ ] = Tells the compiler that we are dealing with an array

Dimension of Array

The dimension denotes the size of array that is classified as one or two or more dimension according to the number of subscript using on it.

One dimensional array

Array whose dimension is classified by one subscript are called single dimension array.

For example: int array[5];

Two dimensional array

Array whose elements are specified by two subscripts are called 2D array. The two dimensional array is called Matrix in which first array declare the number of rows and second array declare the number of columns in the matrix.

For example: int array[6][3];

One Dimension Array

The array is initialized by assigning values to the array element. The value is enclosed in braces and separated by commas. The values must appear in the same order in which they will be assigned to the individual array elements. The array starts from 0 to n-1, where n is the maximum size of the array declared.

The general format of the array initialization is:

data_type array_name[size]={element1, element2,……elementN};

For example:

int values[3]={10,11,12}; or int values[]={10,11,13};

where,

values[0]=10

values[1]=11

values[2]=12

Example1: Write a program to read 10 numbers in an array and display it.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
        int arr[10],i;
        printf("enter any 10 numbers");
        for(i=0;i<=9;i++)
        {
                scanf("%d",&arr[i]);
        }
        printf("\n The required array is:");
        for(i=0;i<=9;i++)
        {
                printf(" %d\t ",arr[i]);
        }
        getch();
}
```

Example2: Write a program to create reverse list of 10 elements in an array.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
    int arr[10], i;
    printf("enter any 10 numbers");
    for(i=0;i<=9;i++)
        scanf("%d", &arr[i]);
    printf("\n the array in reverse order is:");
    for(i=9;i>=0;i--)
        printf("%d \t", arr[i]);
    getch();
}
```

Example3: Write a program to read n numbers in an array and find the largest and smallest number.

```
#include<stdio.h>
#include<conio.h>
#define n 50
void main( )
{
    int x[n], i, large,small;
    for(i=0;i<=n-1;i++)
    {
        scanf("%d", &x[i]);
    }
    large=x[0];
    small=x[0];
    for(i=1;i<=n-1;i++)
    {
        if(x[i]>large)
        {
            large=x[i];
        }
        if(x[i]<small)
        {
            small=x[i];
        }
    }
    printf("\n the largest number is %d \n the smallest value is %d", large,small);
    getch();
}
```

Example4: Write a program to read 5 integer numbers in an array and sort it in ascending order.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int arr[5]={10,2,13,8,7},i,j,temp;
    for(i=0;i<=3;i++)
    {
        for(j=i+1;j<=4;j++)
        {
            if(arr[i]>arr[j])
            {
                temp=arr[i];
```

```
                        arr[i]=arr[j];
                        arr[j]=temp;
                }
        }
    }
    printf("\n the array after sort is:");
    for(i=0;i<=4;i++)
    {
        printf("%4d",arr[i]);
    }
    getch();
}
```

Example5: Write a program to display the Fibonacci series of 10 terms.

```
#include<stdio.h>
#include<conio.h>
void  main( )
{
    Int  fib[10],  i;
    fib[0]=1;
    fib[1]=1;
    for(i=2;i<=9;i++)
        fib[i]=fib[i-1]+fib[i-2];
    printf("\n  the  Fibonacci  series  is");
    printf("%d  \t  %d",  fib[0],  fib[1]);
    for(i=2;i<=9;i++)
        printf("\t  %d  ",  fib[i]);
    getch();
}
```

Example6: Write a program to find whether the given array is palindrome or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[5],i,j=4;
    for(i=0;i<=4;i++)
    {
    scanf("%d",&a[i]);
    }
    i=0;
    while(i<j)
    {
        if(a[i]==a[j])
        {
            i++;
            j--;
        }
        else
        {
        break;
        }
    }
    if(i>=j)
    {
        printf("\ngiven array is palindrome");
    }
    else
    {
```

```
                printf("\ngiven array is not palindrome");
        }
}
```

Example7: WAP to find sum of all prime numbers in a given array.

```
#include<stdio.h>
#include<conio.h>
#define n 10
void main()
{
        int arr[n],i,j,sum=0;
        //to read n array elements
        for(i=0;i<=n-1;i++)
        {
                scanf("%d",&arr[i]);
        }
        for(i=0;i<=n-1;i++)
        {
                for(j=2;j<=arr[i]-1;j++)
                {
                        if(arr[i]%j==0)
                        {
                                break;
                        }
                }
                if(j==arr[i])
                {
                        sum=sum+arr[i];
                }
        }
        printf("\n the sum of all the prime numbers in an array =%d",sum);
}
```

Multidimensional Array

Multidimensional arrays are defined in the same manner as one-dimensional arrays, except that a separate pair of square brackets is required for each subscript.
Thus, a two-dimensional array requires two pairs of square brackets; a three-dimensional array will require three pairs of square brackets and so on.
Two-dimensional array can be expressed as:

```
        data_type  array_name  [rowsize][columnsize];
```

For example:

```
int  x[2][2]={{1,2},{3,4}};
```

where x is a two dimensional array having 2 rows and 2 columns:

```
        x[0][0]=1
        x[0][1]=2
        x[1][0]=3
        x[1][1]=4
```

Example1: Write a program to read a 3x3 matrix and display it on the screen.

```
#include<stdio.h>
#include<conio.h>
void  main( )
{
```

```
	int x[3][3], i, j;
	printf("enter 9 elements");
	//to read array elements
	for(i=0;i<=2;i++)
	{
		for(j=0;j<=2;j++)
		{
			scanf("%d", &x[i][j]);
		}
	}
	//to display array elements
	for(i=0;i<=2;i++)
	{
		for(j=0;j<=2;j++)
		{
			printf("%d\t", x[i][j]);
		}
		printf("\n");
	}
	getch();
}
```

Example2: Write a program to add 3*3.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
	int x[3][3]={{1,2,3},{4,5,6},{7,8,9}}, y[3][3]={{1,1,1},{2,2,2},{3,3,3}},z[3][3], i, j;
	//to add two 3*3 matrix
	for(i=0;i<=2;i++)
	{
		for(j=0;j<=2;j++)
		{
			z[i][j] = x[i][j]+y[i][j];
		}
	}
	printf("\n the matrix after addition is\n");
	for(i=0;i<=2;i++)
	{
		for(j=0;j<=2;j++)
		{
			printf("%d\t", z[i][j]);
		}
		printf("\n");
	}
	getch();
}
```

Example3: Write a program to multiply two 3x3 matrices.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
int x[3][3], y[3][3], z[3][3], i, j, k; /*input*/
printf("enter 9 elements of first matrix");
for(i=0;i<=2;i++)
{
	for(j=0;j<=2;j++)
```

```
        {
                scanf("%d",&x[i][j]);
        }
}
printf("enter 9 elements of second matrix");
for(i=0;i<=2;i++)
{
        for(j=0;j<=2;j++)
        {
                scanf("%d",&y[i][j]);
        }
}
//to multiply two matrix
for(i=0;i<=2;i++)
{
        for(j=0;j<=2;j++)
        {
        z[i][j] = 0;
        for(k=0;k<=2;k++)
        {
                z[i][j]=z[i][j]+x[i][k]*y[k][j];
        }
}
}
printf("\n the resultant matrix is:\n");
for(i=0;i<=2;i++)
{
for(j=0;j<=2;j++)
{
printf("%d\t", z[i][j]);
}
printf("\n");
}
getch();
}
```

Example4: Write a program to find the norm of a matrix. (Norm of a matrix is the square root of sum of square of individual elements.)

```
    #include<stdio.h>
    #include<conio.h>
    #include<math.h>
    void main( )
    {
    int x[3][3], i, j, sum=0, norm;
    printf("enter 9 elements");
    for(i=0;i<=2;i++)
    {
        for(j=0;j<=2;j++)
        {
                scanf("%d", &x[i][j]);
        }
    }
    for(i=0;i<=2;i++) /*find sum of square*/
    {
        for(j=0;j<=2;j++)
        {
                sum=sum+x[i][j]*x[i][j];
```

```
        }
    }
    norm=sqrt(sum); /*find  norm*/
    printf("\n  the  norm  of  the  matrix  is  %d",  norm);
    getch();
    }
    }
```

Example5: Write a program to find the transpose of a 4*5 matrix. (Exam 2010 fall)

```
#include<stdio.h>
#include<conio.h>
void  main( )
{
int  x[4][5], i,  j;
printf("enter  20  elements");
for(i=0;i<=3;i++)
{
        for(j=0;j<=4;j++)
        {
                scanf("%d", &x[i][j]);
        }
}
for(i=0;i<=4;i++)       /*transpose*/
{
        for(j=0;j<=3;j++)
        {
                printf("%d  \t",  x[j][i]);
        }
        printf("\n");
}
getch();
}
```

Example6: WAP to find sum of even elements of 3*3 matrix.(Exam 2011 spring)

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int arr[3][3]={{1,2,3},{4,5,6},{7,8,9}}, i,j,sum=0;
        for(i=0;i<=2;i++)
        {
                for(j=0;j<=2;j++)
                {
                        if(arr[i][j]%2==0)
                        {
                                sum=sum+arr[i][j];
                        }
                }
        }
        printf("\n the sum of even elements of 3*3 matrix is:%d",sum);
getch();
}
```

Example7: WAP to find the average of all the elements of m*n matrix(Exam 2010 spring)

```
#include<stdio.h>
#include<conio.h>
#define m 3
#define n 4
```

```
void main()
{
	int arr[m][n],i,j,sum=0;
	float avg;
	for(i=0;i<=m-1;i++)
	{
		for(j=0;j<=n-1;j++)
		{
			scanf("%d",&arr[i][j]);
		}
	}
	for(i=0;i<=m-1;i++)
	{
		for(j=0;j<=n-1;j++)
		{
		sum=sum+arr[i][j];
		}
	}
	avg=sum/12;
	printf("the sum=%d and average=%f",sum,avg);
	getch();
}
```

Example8: WAP to input m*n matrix and convert it into upper triangular matrix and display it. (Exam 2012 spring)

```
#include<stdio.h>
#include<conio.h>
#define m 4
#define n 4
void main()
{
	int arr[m][n],i,j;
	printf("enter the elemnets in the matrix:\n");
	for(i=0;i<m;i++)
	{
		for(j=0;j<n;j++)
		{
			scanf("%d",&arr[i][j]);
		}
	}
	for(i=1;i<m;i++)
	{
		for(j=0;j<i;j++)
		{
		arr[i][j]=0;
		}
	}
	printf("\n the upper triangular matrix is:\n");
	for(i=0;i<m;i++)
	{
		for(j=0;j<n;j++)
		{
			printf("%d\t",arr[i][j]);
		}
		printf("\n");
	}
	getch();
```

```
        clrscr();
}
```

Example 9: WAP to find the sum of diagonal elements from the left (the trace of a matrix). (Exam 2008 fall)

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int matrix[3][3],sum=0,i,j;
        printf("enter the 3*3 matrix:\n");
        for(i=0;i<=2;i++)
        {
                for(j=0;j<=2;j++)
                {
                        scanf("%d",&matrix[i][j]);
                }
        }
        for(i=0;i<=2;i++)
        {
                for(j=0;j<=2;j++)
                {
                        if(i==j)
                        {
                        sum=sum+matrix[i][j];
                        }
                }
        }
printf("\n the sum of diagonal element of 3*3 matrix is:%d",sum);
getch();
clrscr();
}
```

Character Array or Strings

Strings are sequence of characters used in programming language for storing and manipulating texts such as words, sentence, names etc. In C, there is no built-in data type for strings. String can be represented as a one-dimensional array of characters. A character string is stored in an array of **character** type. String should always have a NULL character (‘\0’) at the end, to represent the end of string.

Initialization of strings

Characters are enclosed within single quotes. String can be initialized in two forms as:

```
char  str[10]={'h','e','l','l','o','\0'};
 OR
char  str[10]="hello";
```

The elements would be assigned as follows.

```
str[0]=‘h’
str[1]=‘e’
str[2]=‘l’
str[3]=‘l’
str[4]=‘o’
str[5]=‘\0’
```

The elements of the string are stored in continuous memory locations. When we initialize the string, the compiler automatically puts a null character ('\0') at the end of last character of the array. The '\0' represents the end of the string. We can declare it as one or multi-dimensional array.

A simple example of string

```
#include <stdio.h>
#include <conio.h>
void main()
{
        char A[20];
        printf("Enter any string\n");
        scanf("%s", A);
        printf("%s", A);
        getch();
}
```

getchar(),putchar(),puts() and gets() function

The getchar() function is used to read a single character from the keyboard and putchar() is used to display the single character on the screen. The execution of scanf() and printf() functions terminates if it found the whitespaces like new line, tab, enter, space etc. So, gets() and puts() are used in place of scanf() and printf() function respectively. The gets() reads a string from the keyboard and puts display the string on the screen. For using these functions, the header file <string.h> is used.

Example of getchar() and putchar()

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
char ch;
ch=getchar();
putchar(ch);
getch();
clrscr();
}
```

Example of gets() and puts()

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char str[10];
        gets(str);
        puts(str);
        getch();

}
```

String Functions

There are various built in functions related to the string. They are available in string.h header file. Some of them are:

String Functions	Functionality
strlen (string)	Gives the length of a string
strcpy(string1, string2)	Copy string2 into string1
strncpy(string1,string2, n)	Copy first n characters of string2 to string1
strcat(string1, string2)	Concatenate string2 at the end of string1
strncat(string1, string2, n)	Append n characters from string2 to string1
strcmp(string1,string2)	It compares two input strings. It returns 0 if string1 is equal to string2, -1 if string1 is less than string2 alphabetically and +1 if string1 is greater than string2 alphabetically. **For example:** If string1[]= “ram” and string2[]= “ram”, then result is 0. If string1[]=“there” and string2 []=“their”, then result is 1. If string1[]=“their” and string2[]= “there”, then result is -1.
strncmp(string1, string2, n)	Compare first n characters of two strings.
strrev (string)	Reverse the string and result is stored in same string.
strupr (string)	Converts string to uppercase and result is stored in same string.
strlwr (string)	Converts a string to lowercase and result is stored in same string.

Example1. WAP to reverse a string entered from keyboard using in build string function.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
char A[10];
printf("Enter a string:");
gets(A);
puts(strrev(A));
getch();
clrscr();
}
```

Example2. WAP to reverse a string entered from keyboard without using strrev() function

```
#include <stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
int i, len;
char A[10];
printf("Enter a string:");
scanf("%s",A);
len=strlen(A);
```

```
for (i=len-1; i>=0;i--)
{
        printf("%c",  A[i]);
}
getch();
clrscr();
}
```

Example3. WAP to copy string from one array onto another using in build string function

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void  main()
{
char  A[10],B[10];
printf("Enter  a  string:");
scanf(" %s",A);
strcpy(B,A);
printf("the content of B is: %s", B);
getch();
clrscr();
}
```

Example4. A program to calculate the length of the string entered from keyboard.

```
#include  <stdio.h>
#include  <conio.h>
#include  <string.h>
void  main()
{
int  len;
char  A[10];
printf("Enter  a  string:");
gets(A);
len=strlen(A);
printf("%d", len);
getch();
clrscr();
}
```

Example5. WAP to concatenate two strings inputted through keyboard. The result of concatenation should be displayed after copying on third variable.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void  main()
{
char  A[10],B[10],C[20];
printf("Enter  first  string:");
gets(A);
printf("Enter  second  string:");
```

```
gets(B);
strcpy(C,  strcat(A,B));
puts(C);
getch();

}
```

Example6. WAP to read a string and rewrite it in alphabetical order.

```
#include <stdio.h>
#include<conio.h>
#include<string.h>
void  main()
{
char  A[10];
int  i,  j,  l,  temp;
printf("Enter  string:");
gets(A);
l  =  strlen(A);
for(i=0;  i<=l-2; i++)
{
        for(j=i+1; j<=(l-1); j++)
        {
        if(A[i]>A[j])
        {
                temp  =  A[i];
                A[i]  =  A[j];
                A[j]  =  temp;
        }
}
}
puts(A);
getch();
}
```

Example7. Print the following pattern (Exam 2009 fall)

```
1N
2NE
3NEP
4NEPA
5NEPAL
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char str[10]="NEPAL";
        int i,j;
        for(i=0;i<strlen(str);i++)
        {
                printf("%d",i+1);
```

```
                for(j=0;j<=i;j++)
                {
                        printf("%c",str[i]);
                }
                printf("\n");
        }
    getch();
    clrscr();
    }
```

Example8. Program to print ENGINEERING as: (Exam 2012 spring)

```
E
EN
ENG
ENGI
ENGIN
ENGINE
ENGINEE
ENGINEER
ENGINEERI
ENGINEERIN
ENGINEERING
```

```
     #include<stdio.h>
    #include<conio.h>
    #include<string.h>
    void main()
    {
        int i,j;
        char n[20]="ENGINEERING";
        for(i=0;i<strlen(n);i++)
        {
            for(j=0;j<=i;j++)
            {
                printf("%c",n[j]);
            }
            printf("\n");
        }
        getch();
    }
```

Q: Why an array called derived data type?

An array is called derived data type because it is derived by using some other data type especially fundamental (basic) data type. It represents a list of elements of an already existing data type.

Q: Explain the difference between 0,'0','\0' and "0".

Here, 0 is an integer value representing zero, '0' is a character value, '\0' is an escape sequence representing null character and "0" is a string representing 0\0 i.e. zero followed by a null character.