KTU NOTES

The learning companion.

KTU STUDY MATERIALS | SYLLABUS | LIVE NOTIFICATIONS | SOLVED QUESTION PAPERS

Website: www.ktunotes.in

MODULE III

Arrays and strings-Arrays Declaration and Initialization, 1-Dimensional Array, 2-Dimensional Array, String processing: In built String handling functions (strlen, strcpy, strcat and strcmp, puts, gets), Linear search program, bubble sort program, simple programs covering arrays and strings

ARRAY

An array in C is a collection of items stored at contiguous memory locations and elements can be accessed randomly using indices of an array. They are used to store similar type (homogeneous) of elements as the data type must be the same for all elements. They can be used to store collection of primitive data types such as int, float, double, char, etc of any particular type.

A	Indices	0	1	2	3
	Array elements	10	20	30	40

Why do we need arrays?

We can use normal variables (v1, v2, v3) when we have a small number of objects, but if we want to store a large number of instances, it becomes difficult to manage them with normal variables. The idea of an array is to represent many instances in one variable.

Different ways of declaring an Array

1. Array declaration by specifying size
   Eg: int A[10];
2. Array declaration by initializing elements
   Eg: int A[] = { 10, 20, 30, 40 };
3. Array declaration by specifying size and initializing elements
   Eg: int A[6] = { 10, 20, 30, 40 };

Advantages of an Array

1. Random access of elements using array index.
2. Use of less line of code as it creates a single array of multiple elements.
3. Easy access to all the elements.
4. Traversal through the array becomes easy using a single loop.
5. Sorting becomes easy as it can be accomplished by writing less line of code.

Disadvantages of an Array

• It allows us to enter only fixed number of elements into it. We cannot alter the size of the array once array is declared. Hence if we need to insert more number of records than declared then it is not possible. We should know array size at the compile time itself.

• Inserting and deleting the records from the array would be costly since we add / delete the elements from the array, we need to manage memory space too.

• It does not verify the indexes while compiling the array. In case there is any indexes pointed which is more than the dimension specified, then we will get run time errors rather than identifying them at compile time.

Example:

```
int a[3]={1,2,3}
printf("%d",A[3]);
```

Output: 3245431 (Garbage Value)

1. Write a program to read and display an array of size n.

```
#include<stdio.h>
void main()
{
        int n,A[20],i;
        printf("Enter the size of Array:");
        scanf("%d",&n);
         for(i=0;i<n;i++)
        {
                printf("\nEnter the number:");
                scanf("%d",&A[i]);
        }
        for(i=0;i<n;i++)
        {
                printf("%d\t",A[i]);
        }
}
```

OUTPUT

```
Enter the size of Array: 5
Enter the number:10
Enter the number:20
Enter the number:30
Enter the number:40
Enter the number:50
```

```
10      20      30      40      50
```

2. Write a program to read and display an array of size n in reverse order.

```
#include<stdio.h>
void main()
{
        int n, A[20],i;
        printf("Enter the size of Array:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("\nEnter the number");
                scanf("%d",&A[i]);
        }
        for(i=n-1;i>=0;i--)
        {
                printf("%d\t",A[i]);
        }
}
```

OUTPUT

```
Enter the size of Array: 5
Enter the number:10
Enter the number:20
Enter the number:30
Enter the number:40
Enter the number:50
50      40      30      20      10
```

3. Write a program to find sum and average of an array of size n.

```
#include<stdio.h>
void main()
{
        int n,A[20],i,sum; float avg;
        printf("Enter the size of Array:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("\nEnter the number");
                scanf("%d",&A[i]);
        }
        sum=0;
        for(i=0;i<n;i++)
```

```
        {
                sum = sum + A[i];
        }
        avg = (float)sum/n;
        printf("\nSum=%d",sum);
        printf("\nAverage=%f",avg);
}
```

OUTPUT

```
Enter the size of Array: 5
Enter the number:10
Enter the number:20
Enter the number:30
Enter the number:40
Enter the number:50
Sum=150
Average=30.0
```

4. Write a program to read and display even numbers in an array of size n.

```
#include<stdio.h>
void main()
{
        int n,A[20],i;
        printf("Enter the size of Array:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("\nEnter the number:");
                scanf("%d",&A[i]);
        }
        printf("\nEven Numbers are\t");
        for(i=0;i<n;i++)
        {
                if(A[i]%2 == 0)
                        printf("%d\t",A[i]);
        }
}
```

OUTPUT

```
Enter the size of Array: 5
Enter the number:10
Enter the number:15
Enter the number:20
Enter the number:25
```

```
Enter the number:30
Even Numbers are 10          20       30
```

5. Write a program to read and display prime numbers in an array of size n.

```
#include<stdio.h>
#include<math.h>
void main()
{
        int n,A[20],i,flag,j;
        printf("Enter the size of Array:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("\nEnter the number:");
                scanf("%d",&A[i]);
        }
        printf("\nPrime Numbers are\t");
        for(i=0;i<n;i++)
        {
                flag=0;
                for(j=2;j<=sqrt(A[i]);j++)
                {
                        if(A[i]%j == 0)
                        {
                                flag=1;
                                break;
                        }
                }
                if(flag == 0)
                {
                        printf("%d\t",A[i]);
                }
        }
}
```

OUTPUT

```
Enter the size of Array: 5
Enter the number:5
Enter the number:6
Enter the number:7
Enter the number:8
Enter the number:9
Prime Numbers are   5        7
```

6. Write a program to find largest element in an array of size n.

```
#include<stdio.h>
void main()
{
        int n,A[20],i,large;
        printf("Enter the size of Array:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
        printf("\nEnter the number");
        scanf("%d",&A[i]);
        }
        large=A[0];
        for(i=1;i<n;i++)
        {
                if(large < A[i])
                        large = A[i]
        }
        printf("\nLarge =%d",large);
}
```


OUTPUT

```
Enter the size of Array: 5
Enter the number:20
Enter the number:30
Enter the number:25
Enter the number:15
Enter the number:50
Large=50
```

LINEAR SEARCH (SEQUENTIAL SEARCH)

Searching is an operation or a technique that helps finds the place of a given element or value in the list. Any search is said to be successful or unsuccessful depending upon whether the element that is being searched is found or not.

Linear search is a very basic and simple search algorithm. In Linear search, we search an element or value in a given array by traversing the array from the starting, till the desired element or value is found. It compares the element to be searched with all the elements present in the array and when the element is matched successfully, it returns the index of the element in the array, else it return -1.

Linear Search is applied on unsorted or unordered lists, when there are fewer elements in a list.

Algorithm

Linear Search (Array A, Value x)

Step 1: Set i to 0

Step 2: if i = n-1 then go to step 7

Step 3: if A[i] = x then go to step 6

Step 4: Set i to i + 1

Step 5: Go to Step 2

Step 6: Print Element x Found at index i and go to step 8

Step 7: Print element not found

Step 8: Exit

Write a program to search an element in an array of size n.(linear search)

```
#include<stdio.h>
void main()
{
        int n,A[20],i,pos=-1;
        printf("Enter the size of Array:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("\nEnter the number:");
                scanf("%d",&A[i]);
        }
        printf("\nEnter the element to search");
        scanf("%d",&key);
        for(i=0;i<n;i++)
        {
                if(A[i] == key)
                {
                        pos= i;
                        break;
                }
        }
        if(pos == -1)
        {
                printf("\nElement not found");
        }
        else
```

```
        {
                prinf("\nElement found at index %d",pos);
        }
}
```

OUTPUT

```
Enter the size of Array: 5
Enter the number:10
Enter the number:20
Enter the number:30
Enter the number:40
Enter the number:50
Enter the element to search:40
Element found at index 3
```

BUBBLE SORT

A sorting algorithm is an algorithm that puts elements of a list in a certain order. The bubble sort makes multiple passes through a list. It compares adjacent items and exchanges those that are out of order. Each pass through the list places the next largest value in its proper place. In essence, each item "bubbles" up to the location where it belongs.

Example:

Arrange the following elements in ascending order

Pass 1

9	6	5	2
6	9	5	2
6	5	9	2
6	5	2	9

Pass 2

6	5	2	9
5	6	2	9
5	2	6	9

Pass 3

5	2	6	9
2	5	6	9

Write a program to sort the elements in an array of size n.(BUBBLE SORT)

```
#include<stdio.h>
void main()
{
	int n,A[20],i,j,temp;
	printf("Enter the size of Array");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("\nEnter the number:");
		scanf("%d",&A[i]);
	}
	printf("Before Sorting");
	for(i=0;i<n;i++)
	{
		printf("%d\t",A[i]);
	}
	for(i=0;i<n-1;i++)
	{
		for(j=0;j<n-i-1;j++)
		{
			if(A[j]>A[j+1])
			{
				temp=A[j];
				A[j]=A[j+1];
				A[j+1]=temp;
			}
		}
	}
	printf("\nAfter Sorting \t");
	for(i=0;i<n;i++)
	{
		printf("%d\t",A[i]);
	}
}
```


STRINGS

Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character '\0'.

Index	0	1	2	3	4	5
Character Array	'H'	'E'	'L'	'L'	'O'	

String	'H'	'E'	'L'	'L'	'O'	'\0'

Declaration of Strings

Declaring a string is as simple as declaring a one dimensional array. Below is the basic syntax for declaring a string.

char str[20];

In the above syntax str is any name given to the string variable and 20 is used define the length of the string, i.e. the number of characters strings will store(including null character). Please keep in mind that there is an extra terminating character which is the Null character ('\0') used to indicate termination of string which differs strings from normal character arrays.

Initializing a String

A string can be initialized in different ways.

1. char str[] = "Programming";
2. char str[50] = "Programming";
3. char str[] = {'P','r','o','g','r','a','m','m','i','n','g','\0'};
4. char str[12] = {'P','r','o','g','r','a','m','m','i','n','g','\0'};

Read a character from User

1. scanf()

- It is used to read the input (character, string, numeric data) from the standard input (keyboard).
- It is used to read the input until it encounters a whitespace, newline or End Of File (EOF).

```
#include <stdio.h>
void main()
{
        char str[20];
        printf("Enter the String: ");
        scanf("%s", str);
        printf("\nString: %s\n", str);
}
```

Output

Enter the String : Hello World

String: Hello

Issue with scanf(): There is a whitespace after Hello. So it read the input till Hello and store it in str.

Solution: use % [^\n] instead of %s.

```
#include <stdio.h>
void main()
{
        char str[20];
        printf("Enter the String:");
        scanf("%[^\n]", str);
        printf("\nString: %s\n", str);
}
```

Output

Enter the String : Hello World

String: Hello World

2. gets()

- It is used to read input from the standard input (keyboard).
- It is used to read the input until it encounters newline or End Of File (EOF).

```
#include <stdio.h>
void main()
{
        char str[20];
        printf("Enter the String:");
        gets( str);
        printf("\nString: %s\n", str);
}
```


Output

Enter the String: Hello World

String: Hello World

Issue: gets() reads string from standard input and prints the entered string, but it suffers from Buffer Overflow as gets() doesn't do any array bound testing. gets() keeps on reading until it sees a newline character.

Display String value – printf and puts

```
#include <stdio.h>
void main()
{
        char str[20];
```

```
        printf("Enter the String:");
        gets( str);
        printf("%s\n", str);
}
```

Output

Enter the String: Hello world Hello world

```
#include <stdio.h>
void main()
{
        char str[20];
        printf("Enter the String:");
        gets( str);
        puts(str);
}
```

Output

Enter the String: Hello world Hello world

IN-BUILT STRING FUNCTIONS

Function	Working
strlen()	computes string's length
strcpy()	copies a string to another
strcat()	concatenates(joins) two strings
strcmp()	compares two strings case sensitive checking
strcmpi()	compares two strings case insensitive checking
strlwr()	converts string to lowercase
strupr()	converts string to uppercase

strlen() vs sizeof()

strlen returns the length of the string stored in array, however sizeof returns the total allocated size assigned to the array.

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

```
char str[]="Hello World";
int len=strlen(str);
int size=sizeof(str);
printf("Value=%d",len*size);
}
```

len value will the length of the string which is 11. But size value will be 12 including the end of character.

Output

Value=132

strcat

It concatenates two strings and returns the concatenated string.

Syntax: strcat(string1,string2) – concatenate string1 with string2.

```
#include <stdio.h>
#include <string.h>
void main()
{
char s1[10] = "Hello";
char s2[10] = "World";
strcat(s1,s2);
printf("After concatenation: %s", s1);
}
```

Output:

After concatenation: HelloWorld

Program to concatenate two strings without using string handling function

```
#include <stdio.h>
void main()
{
        char str1[50], str2[25], i, j;
        printf("\nEnter first string: ");
        gets(str1);
        printf("\nEnter second string: ");
        gets(str2);
        for(i=0; str1[i]!='\0'; i++);
        for(j=0; str2[j]!='\0'; j++, i++)
        {
```

```
            str1[i]=str2[j];
        }
        str1[i]='\0';
        printf("\nOutput: %s",str1);
}
```

strcpy

It copies the string str2 into string str1, including the end character (terminator char '\0').

Syntax: strcpy(string1,string2) - copy the content of string2 to string1

```
#include <stdio.h>
#include <string.h>
void main()
 {
        char s1[30];
        char s2[30] = "Hello World";
        strcpy(s1,s2);
        printf("String s1: %s", s1);
}
```

Output:

String s1 : Hello World

Program to copy a string to another without using string handling function

```
#include<stdio.h>
void main()
{
        char a[50], b[50];
        int i=0;
        printf("Enter string1:");
        gets(a);
        for(i=0;a[i]!='\0';i++)
        {
                b[i]=a[i];
         }
        b[i]='\0';
}
```

strcmp

It compares the two strings and returns an integer value. If both the strings are same (equal) then this function would return 0 otherwise it may return a negative or positive value based on the comparison.

strcmp(string1,string2) -- Compares the content of string1 and string2

•If string1 < string2 OR string1 is a substring of string2 then it would result in a negative value.

•If string1 > string2 then it would return positive value.

•If string1 == string2 then you would get 0(zero) when you use this function for compare strings.

```
#include <stdio.h>
#include <string.h>
void main()
{
        char s1[20] = "Hello";
        char s2[20] = "World";
        if (strcmp(s1, s2) ==0)
        {
                printf("string 1 and string 2 are equal");
        }
        else
        {
                printf("string 1 and 2 are different");
        }
}
```

Output

string 1 and 2 are different

Program to check two strings are equal without using string handling functions

```
#include<stdio.h>
void main()
{
        char str1[100], str2[100];
        int i, flag=0;
        printf("Enter the First string :\n");
        gets(str1);
        printf("Enter the Second string :\n");
        gets(str2);
        for(i=0;str1[i]!='\0'&&str2[i]!='\0';i++)
        {
                if(str1[i]!=str2[i])
```

```
            {
                flag=1;
                break;
            }
        }
        if(flag==0&& str1[i]!='\0'&&str2[i]!='\0')
            printf("Strings are equal \n");
        else
            printf("Strings are not equal \n");
}
```

1. Write a program to display the content of string in upper and lower case.

```
#include<stdio.h>
#include<string.h>
void main()
{
    char str[20];
    printf("Enter the string:");
    gets(str);
    printf("\nUpper Case String:%s",strupr(str));
    printf("\nLower Case String:%s",strlwr(str));
}
```


Output

Enter the string: Hello World
Upper Case String: HELLO WORLD
Lower Case String: hello world

2. Write a program to reverse a string.

```
#include<stdio.h>
#include<string.h>
void main()
{
    char str[20];
    int len,i,j;
    printf("Enter the string:");
    gets(str);
    i=strlen(str)-1;
    j=0;
    while(i>=0)
    {
```

```
                rev[j]=str[i];
                j++;
                i--;
        }
        rev[j]='\0';
        printf("\nReverse=%s",rev);
}
```

Output

```
Enter the string: hello
Reverse=olleh
```

3. Write a program to check whether a string is palindrome or not without using in built functions.

```
#include<stdio.h>
#include<string.h>
void main()
{
        char str[20];
        int len,i,j,flag;
        printf("Enter the string:");
        gets(str);
        // Code to find string length
        for(len=0;str[len]!='\0';len++);
        // Code to check the Palindrome
        flag=0;
        for(i=0;i<len/2;i++)
        {
                if(str[i] != str[len-i-1])
                {
                        flag=1;
                        break;
                }
        }
        if(flag == 0)
        {
                printf("\nPalindrome");
        }
        else
        {
                printf("\nNot Palindrome");
        }
}
```

Output

Enter the string: malayalam
Palindrome

4. Write a program to count number of vowels in a given string.

```
#include<stdio.h>
#include<string.h>
void main()
{
        char str[20
        int i,count;
        printf("Enter the string:");
        gets(str);
        count=0;
        for(i=0;i<strlen(str);i++)
        {
                switch(str[i])
                {
                        case 'A':
                        case 'a':
                        case 'E':
                        case 'e':
                        case 'i':
                        case 'I':
                        case 'O':
                        case 'o':
                        case 'U':
                        case 'u':
                                        count++;
                                        break;
                }
        }
        printf("\nCount of vowels=%d",count);
}
```

Output

Enter the string:hello
Count of vowels=2

TWO DIMENSIONAL ARRAY

An array of arrays is known as 2D array. The two dimensional (2D) array is also known as matrix. A matrix can be represented as a table of rows and columns. The syntax to declare the 2D array is given below.

data_type array_name[rows][columns];

INITIALIZATION OF TWO DIMENSIONAL ARRAY

There are two ways to initialize a two Dimensional arrays during declaration.

1. int A[2][3] = { {10, 11, 12},{14, 15, 16}};
   int A[][3] = { {10, 11, 12},{14, 15, 16}};
2. int A[2][3] = { 10, 11, 12, 14, 15, 16};
   int A[][3] = { 10, 11, 12, 14, 15, 16};

When we initialize a 1 dimensional array during declaration, we need not to specify the size of it. However that's not the case with 2D array, you must always specify the second dimension even if you are specifying elements during the declaration.

/* Valid declaration*/

int A[2][2] = {1, 2, 3 ,4 }

/* Valid declaration*/

int A[][2] = {1, 2, 3 ,4 }

/* Invalid declaration – you must specify second dimension*/ int A[][] = {1, 2, 3 ,4 }

/* Invalid because of the same reason mentioned above*/ int A[2][] = {1, 2, 3 ,4 }

Structure of two dimensional array of size - A[3][5]

A[0][0]	A[0][1]	A[0][2]	A[0][3]	A[0][4]
A[1][0]	A[1][1]	A[1][2]	A[1][3]	A[1][4]
A[2][0]	A[2][1]	A[2][2]	A[2][3]	A[2][4]

1. Write a program to read a matrix of size mXn.

```
#include<stdio.h>
void main()
{
        int m,n,A[20][20],i,j;
        printf("Enter the order of matrix:");
        scanf("%d%d",&m,&n);
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        printf("Enter the element:");
                        scanf("%d",&A[i][j]);
                }
```

```
        }
        printf("\nMatrix\n");
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        printf("%d\t",A[i][j]);
                }
                printf("\n");
        }
}
```

OUTPUT

```
Enter the order of matrix: 2 2
Enter the element: 10
Enter the element: 20
Enter the element: 30
Enter the element: 40
Matrix
10      20
30      40
```

2. Write a program to read a matrix of size mXn and display its transpose.


```
#include<stdio.h>
void main()
{
        int m,n,A[20][20],i,j;
        printf("Enter the order of matrix:");
        scanf("%d%d",&m,&n);
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        printf("Enter the element:");
                        scanf("%d",&A[i][j]);
                }
        }
        printf("\nTranspose of Matrix\n");
        for(j=0;j<n;j++)
        {
                for(i=0;i<m;i++)
                {
                        printf("%d\t",A[i][j]);
```

```
                }
                printf("\n");
        }
}
```

OUTPUT

```
Enter the order of matrix: 2 2
Enter the element: 10
Enter the element: 20
Enter the element: 30
Enter the element: 40
Transpose of Matrix
10      30
20      40
```

3. Write a program to read a matrix of size mXn and display the sum of principal diagonal elements.

```
#include<stdio.h>
void main()
{
        int m,n,A[20][20],i,j;
        printf("Enter the order of matrix:");
        scanf("%d",&m,&n);
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        printf("Enter the element:");
                        scanf("%d",&A[i][j]);
                }
        }
        for(i=0;i<m;i++)
        {
                sum = sum + A[i][i];
        }
        printf("\nSum=%d",sum);
}
```

OUTPUT

```
Enter the order of matrix: 2 2
Enter the element: 10
Enter the element: 20
Enter the element: 30
Enter the element: 40
```

Sum=50

4. *Write a program to multiply two matrices*

To multiply two matrices, the number of columns of the first matrix should be equal to the number of rows of the second matrix.

```
#include <stdio.h>
void main()
{
	int i,j,k,m, n, p, q,sum = 0;
	int A[10][10], B[10][10], C[10][10];
	printf("Enter number of rows and columns of first matrix\n");
	scanf("%d%d", &m, &n);
	printf("Enter number of rows and columns of second matrix\n");
	scanf("%d%d", &p, &q);
	if (n != p)
		printf("The multiplication isn't possible.\n");
	else
	{
		printf("Enter elements of first matrix\n");
		for (i = 0; i < m; i++)
			for (j = 0; j < n; j++)
				scanf("%d", &A[i][j]);
		printf("Enter elements of second matrix\n");
		for (i = 0; i < p; i++)
			for (j = 0; j < q; j++)
				scanf("%d", &B[i][j]);
		for (i= 0; i < m; i++)
		 {
			for (j = 0; j < q; j++)
			 {
				C[i][j]=0;
				 for (k = 0; k < p; k++)
				{
					C[i][j]= C[i][j] + A[i][k]*B[k][j];
				}
			}
		}
		printf("Product of the matrices:\n");
		for (i = 0; i< m; i++)
		{
			for (j = 0; j< q; j++)
```

```
                {
                        printf("%d\t", C[i][j]);
                }
                printf("\n");
            }
        }
}
```

PREVIOUS YEAR UNIVERSITY QUESTIONS

1. Explain the different ways in which you can declare & initialize a single dimensional array. ***[KTU, MODEL 2020]***
2. Write a C program to read a sentence through keyboard and to display the count of white spaces in the given sentence. ***[KTU, MODEL 2020]***
3. Write a C program to check whether a given matrix is a diagonal matrix. ***[KTU, MODEL 2020]***
4. Write a C program to perform bubble sort. [***KTU, MODEL 2020], [KTU, JULY 2017], [KTU, JULY 2018]***
5. Without using any built in string processing function like strlen, strcat etc., write a program to concatenate two strings. ***[KTU, MODEL 2020]***
6. Explain with an example how multidimensional arrays are declared? [***KTU, DECEMBER 2019]***
7. Illustrate sorting of the following elements using bubble sort (ascending)
   21, 32, 45, 14,67,8, 54, 9 ***[KTU, DECEMBER 2019]***
8. Write a C program to subtract two matrices. ***[KTU, MAY 2019]***
9. Write a C program to accept a two dimensional matrix and display the row sum, column sum and diagonal sum of elements. ***[KTU, MAY 2019]***
10. Write a C program to replace a character in a string with another character. ***[KTU, MAY 2019]***
11. Write a C program to read two sorted arrays and merge them into a single array. ***[KTU, MAY 2019]***
12. Write a C program to concatenate two strings without using any standard library functions ***[KTU, MAY 2017], [KTU, JULY 2017], [KTU, DECEMBER 2018]***
13. Give the declaration of variable for storing the string "PROGRAMMING" in C. ***[KTU, JULY 2017]***
14. Write a C program to find the largest and smallest numbers and their locations in an array of n numbers. ***[KTU, JULY 2017]***
15. Write a C program to find the transpose of a matrix. [***KTU, JULY 2017], [KTU, JULY 2018], [KTU, DECEMBER 2019]***
16. Write a C program to accept a 2-D integer matrix and check whether it is symmetric or not. ***[KTU, APRIL 2018]***
17. Write a C program to check whether a character is present in a string. ***[KTU, APRIL 2018]***

18. Write a C program for displaying the prime numbers in a mXn matrix. ***[KTU, APRIL 2018]***
19. Write a C program to sort names in an array in lexicographical order. ***[KTU, DECEMBER 2018]***
20. Write a C program to find second largest element of an unsorted array. ***[KTU, DECEMBER 2018], [KTU, DECEMBER 2019]***
21. Given a set of n items, write a C program to find the k^{th} largest item in the list ***[KTU, DECEMBER 2018]***
22. Explain how the string variables are declared and initialised in C program. ***[KTU, JULY 2018]***
23. Write a C program that reads a string from keyboard and determines whether the string is palindrome or not. ***[KTU, JULY 2018]***
24. Give the syntax of four string handling functions in C ***[KTU, JULY 2018].***
25. Write a C program to search a given number from a set of numbers using linear search method. Explain linear search method with example ***[KTU, JULY 2018]***
26. How do you initialize a two dimensional array during declaration? ***[KTU, MAY 2017]***
27. Give the purpose of following functions with examples. i) strcmp() ii) strcat() iii) strupr() ***[KTU, APRIL 2018]***
28. Write a program to find the sum, average and product of a set of N numbers, using arrays. ***[KTU, APRIL 2018]***
29. Write a program to read a matrix, replace all negative elements of the matrix by zero and print the resulting array. ***[KTU, APRIL 2018]***
30. Develop C program to find the string length without using inbuilt functions ***[KTU, JUNE 2017]***
31. Write a program which accepts two 3x3 matrices from the user and print its product. **[KTU, APRIL 2018]**
32. Write a C program to multiply two 5x5 matrices. **[KTU, DECEMBER 2018]**
33. Write a C program to receive 10 numbers in an array and to sort it in ascending order. **[KTU, DECEMBER 2018]**