Module 3
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

Arrays

A variable can store only one value. An array is a collective name given to a group of **similar elements**. All the elements in an array will be of same type. Similar elements could be all int, or all float, or all char etc. An array is a fixed size sequenced collection of elements of the same data type. An array in C is a collection of similar data items stored at **contiguous memory locations and elements can be accessed randomly using indices of an array.**

TYPE OF ARRAYS

- One dimensional arrays
- Two- dimensional arrays
- Multi- dimensional arrays

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.

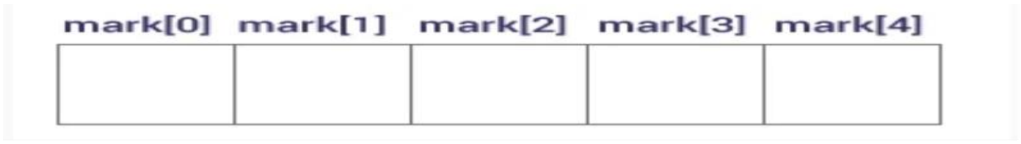
ONE DIMENSIONAL ARRAY

To begin with, like other variables an array needs to be declared so that the compiler will know what kind of an array and how large an array we want.

The general **syntax** is:

type array_ name [size];

Eg : int mark[5];



An array is a collection of similar elements. The first element in the array is in index 0, so the last element index is 1 less than the size of the array. An array is also known as a subscripted variable. The element numbers in [ ] are called index or subscript. Before using an array its type and dimension must be declared. However big an array, its elements are always stored in contiguous memory locations.

Array initialization

- Compile Time Initialization:

  Array declaration by initializing elements

  Eg: int A[] = { 10, 20, 30, 40 } ;

  Array declaration by specifying size and initializing elements

  Eg: int A[6] = { 10, 20, 30, 40 };

- Run Time Initialization:

  Initialization done at runtime using scanf() function. This approach is usually used for initializing large array, or to initialize array with user specified values.

Characteristics of Array

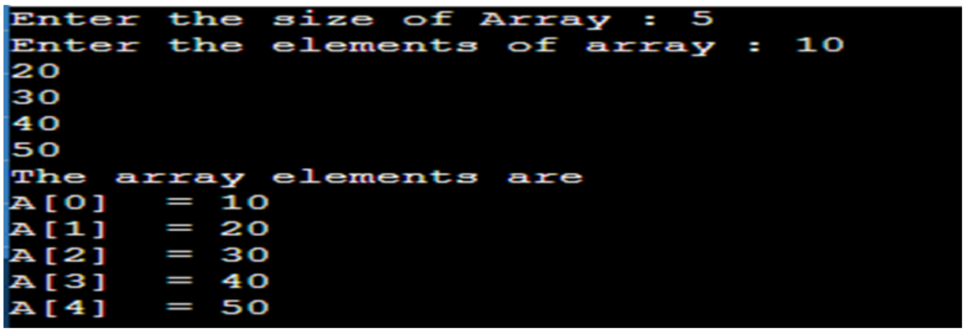
1. Declaration int mark[5] is the creation of 5 integer types in memory.
2. All elements of an array share the same name and they are distinguished with the help of element number.
3. Any particular element in an array can be modified separately without disturbing other elements.
4. Any element of an array a[] can be assigned to another ordinary variable or array variable of its type.
   a. Eg: b=a[2];
   b. a[2]=a[3];
5. Single operation which involves entire array is not allowed in c.

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

```
#include<stdio.h>

int main()

{

            int n,A[20],i;

            printf("Enter the size of Array : ");

            scanf("%d",&n);

            printf("Enter the elements of array : ");

            for(i=0;i<n;i++)

            {

                    scanf("%d",&A[i]);

            }

             printf("The array elements are \n");
```

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

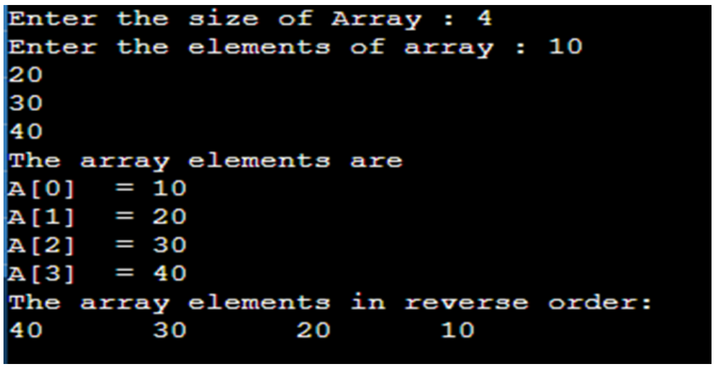
<u>OUTPUT</u>



```
Enter the size of Array : 5
Enter the elements of array : 10
20
30
40
50
The array elements are
A[0]   = 10
A[1]   = 20
A[2]   = 30
A[3]   = 40
A[4]   = 50
```

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

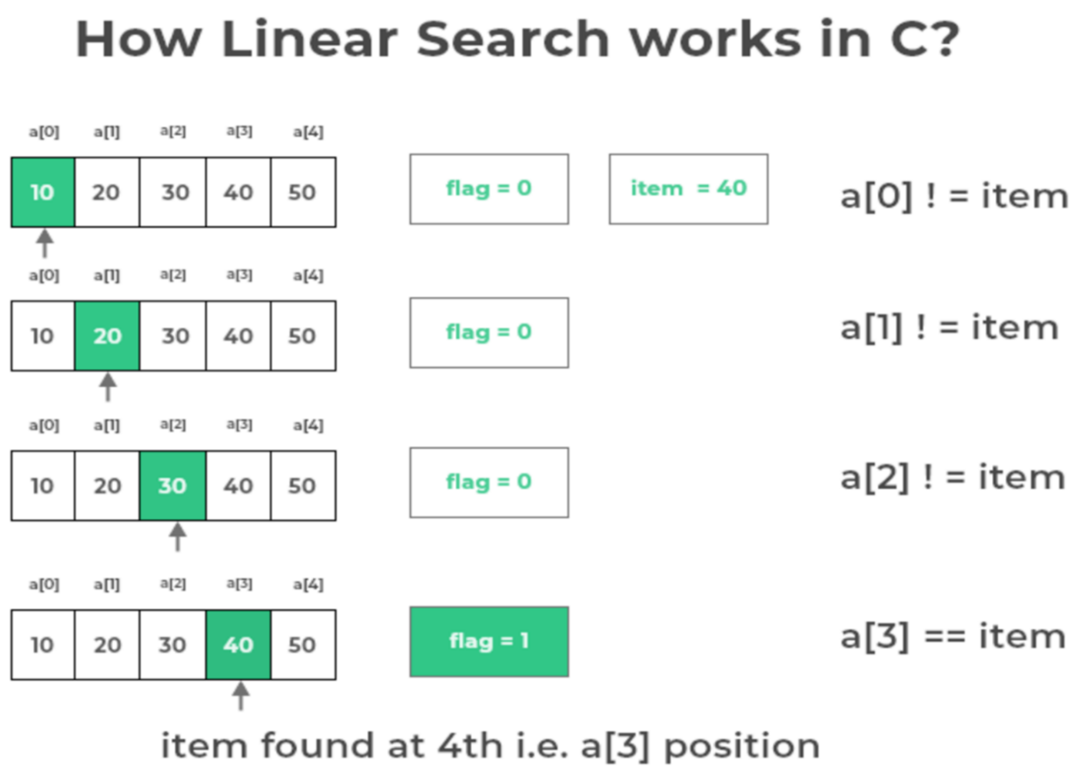
```
#include<stdio.h>
int main()
{
                int n,A[20],i;
                printf("Enter the size of Array : ");
                scanf("%d",&n);
                printf("Enter the elements of array : ");
                for(i=0;i<n;i++)
                {
                  scanf("%d",&A[i]);
  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



```
Enter the size of Array : 4
Enter the elements of array : 10
20
30
40
The array elements are
A[0]   = 10
A[1]   = 20
A[2]   = 30
A[3]   = 40
The array elements in reverse order:
40      30      20      10
```

LINEAR SEARCH

Search is a process of finding a value in a list of values. This search process starts comparing search element with the first element in the list. If both are matched then result is element found otherwise search element is compared with the next element in the list. Repeat the same until search element is compared with the last element in the list, if that last element also doesn't match, then the result is "Element not found in the list". That means, the search element is compared with element by element in the list.



Algorithm – Linear search

```
Step 1:Declare array a, size n, search element item, flag=0,i
Step 2: Read the size of the array n
Step3 : Read the array elements
Step4 : Read the search element item
Step 5: Repeat steps 6 -7  until i < n
Step 6: if a[i] == item then
        Step 6.1: Set flag = 1
        Step 6.2: Print Element found at position i+1
Step 7:i=i+1
Step 8: if flag=0
        Step 8.1 : Print element not found
Step 9: Exit
```

Program – Linear Search

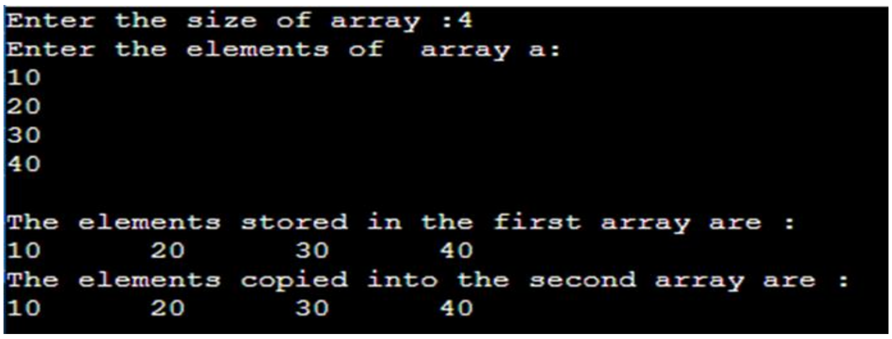
```
#include <stdio.h>              //header files
 int main ()                    //main function
{
            int a[20],n,item,i,flag=0;
            printf("Enter size of the array ");
            scanf("%d",&n);
            printf("Enter array elements");
            for(i = 0; i <n; i++)
            {
                    scanf("%d",&a[i]);
            }
             printf("Enter the element to be Search: ");
            scanf("%d",&item);
            for (i = 0; i<n; i++)             //looping logic
            {
                    if(a[i] == item)
```

```
                {
                        flag = 1;
                         printf("\nItem found at location %d\n",i+1);
                        break;
                 }
        }
         if(flag == 0)
                printf("\nItem not found\n");
}
```

Write a program to copy content of one array to another array.

```
#include <stdio.h>
void main()
{
   int a[20], b[20],i,n;
   printf("Enter the size of array :");
   scanf("%d",&n);

   printf("Enter the elements of  array a:\n");
     for(i=0;i<n;i++)
      {
          scanf("%d",&a[i]);
      }
   /* Copy elements of first array into second array.*/
   for(i=0; i<n; i++)
   {
      b[i] = a[i];
   }
/* Prints the elements of first array   */

   printf("\nThe elements stored in the first array are :\n");
```

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

    /* Prints the elements copied into the second array. */

    printf("\nThe elements copied into the second array are :\n");
    for(i=0; i<n; i++)
    {
        printf("%d\t",b[i]);
    }
}
```

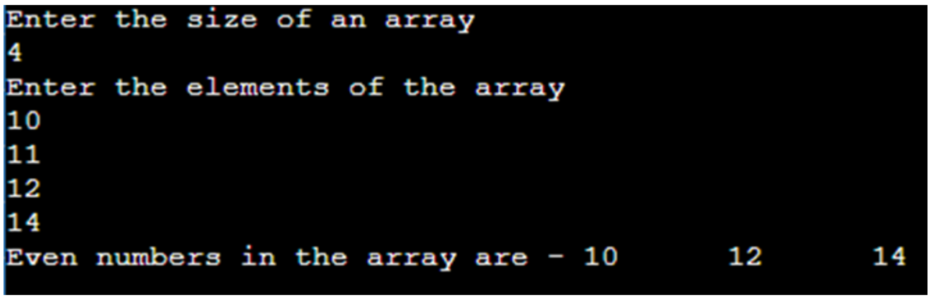


```
Enter the size of array :4
Enter the elements of  array a:
10
20
30
40

The elements stored in the first array are :
10      20      30      40
The elements copied into the second array are :
10      20      30      40
```

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

```
#include <stdio.h>
    void main()
    {
         int array[100], i, num;
        printf("Enter the size of an array \n");
         scanf("%d", &num);
        printf("Enter the elements of the array \n");
         for (i = 0; i < num; i++)
        {
            scanf("%d", &array[i]);
        }
```

```
         printf("Even numbers in the array are - ");
        for (i = 0; i < num; i++)
        {
            if (array[i] % 2 == 0)
            {
                printf("%d \t", array[i]);
            }
        }
    }
```



```
Enter the size of an array
4
Enter the elements of the array
10
11
12
14
Even numbers in the array are - 10      12      14
```

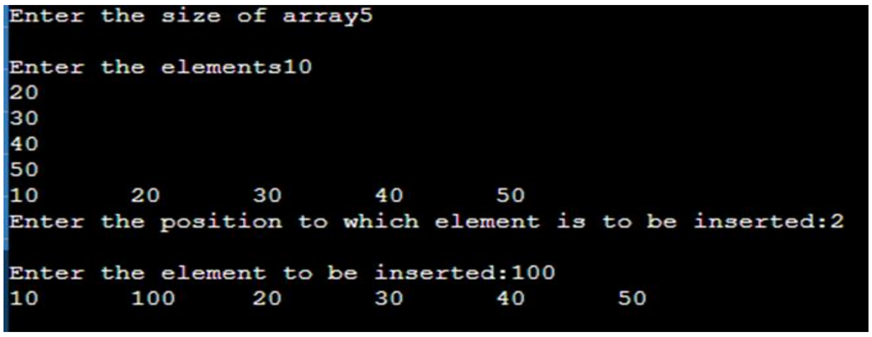
Write a program to find sum and average of an array content of size n.

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

```
                        sum = sum +A[i];
            }
            avg = (float)sum/n;
            printf("Sum=%d",sum);
            printf("Average=%f",avg);
}
```

Write a Program to insert an element at desired position in an array

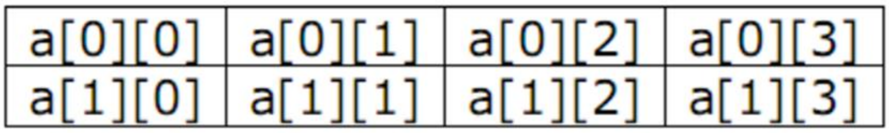
```
#include<stdio.h>
void main()
{
    int arr[50],n,i,pos,element;
    printf("Enter the size of array");
    scanf("%d",&n);
    printf("\nEnter the elements");
    for(i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    for(i=0;i<n;i++)
    {
        printf("%d\t",arr[i]);
    }
printf("\nEnter the position to which element is to be inserted:");
    scanf("%d",&pos);
    printf("\nEnter the element to be inserted:");
 scanf("%d",&element);
for(i=n;i>=pos;i--)
    {
        arr[i]=arr[i-1];
    }
    arr[pos-1]=element;
    for(i=0;i<n+1;i++)
    {
        printf("%d \t",arr[i]);
    }
}
```



```
Enter the size of array5

Enter the elements10
20
30
40
50
10      20      30      40      50
Enter the position to which element is to be inserted:2

Enter the element to be inserted:100
10      100     20      30      40      50
```

TWO DIMENSIONAL ARRAYS

- Data are stored in rows and columns.



a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]

General syntax for declaring an array is:

Datatype arrayname[rowsize][columnsize];

- datatype refers to the type of elements in array.
- arrayaname refers to the name of array
- row size shows the number of rows an array
- columnsize shows the number of columns an array

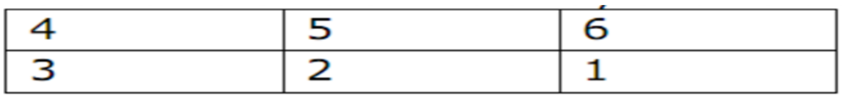
Eg: int a[3][4];

shows an array named a with 3 rows and 4 columns.

Intializing 2D array

1.) Eg: int A[2][3]={4,5,6,3,2,1};

Here array named A consist of 6 elements . Size of the array and values are assigned directly.



4	5	6
3	2	1

2.)Eg: int A[2][3]={{4,5,6},{3,2,1}};

4	5	6
3	2	1

3) Eg: int arr[ ][3] = { 52, 30, 23, 55, 56, 85,20,25,40 } ;

It is important to remember that while initializing a 2-D array it is necessary to mention the second (column) dimension, whereas the first dimension (row) is optional.

4.)Eg: int a[2][3]={{4},{3,2}};

Row elements are kept in ordered set. Since the column size is 3 , remaining fields will be filled with 0

4	0	0
3	2	0

int arr[2][ ] = { 52, 30, 23, 55, 56, 85 } ;

int arr[ ][ ] = { 52, 30, 23, 55, 56, 85 } ;

The above two statements would never work.

int A[2][2] = {1, 2, 3 ,4 };   // Valid declaration

int A[][] = {1, 2, 3 ,4 } ;   // Invalid declaration

int A[2][] = {1, 2, 3 ,4 };   // Invalid declaration

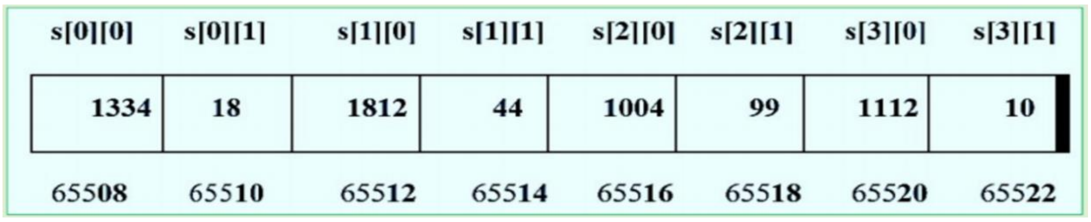
int A[][2] = {1, 2, 3 ,4 };   // Valid declaration

int A[5][4]={10,20,30,40};   // Valid declaration

<u>Memory Map of a 2-Dimensional Array</u>

Memory doesn’t contain rows and columns. In memory whether it is a one-dimensional or a two-dimensional array the array elements are stored in one contiguous chain.

The arrangement of array elements of a two-dimensional array in memory is shown below:



s[0][0]	s[0][1]	s[1][0]	s[1][1]	s[2][0]	s[2][1]	s[3][0]	s[3][1]
1334	18	1812	44	1004	99	1112	10
65508	65510	65512	65514	65516	65518	65520	65522

Write a C program to store elements in 2D array and display it.

```
#include<stdio.h>
int main( )
{
            int a[100][100],i,j,rows,columns;
            printf("enter the number of rows : ");
            scanf("%d",&rows);
            printf("enter the number of columns : ");
             scanf("%d",&columns);
            printf("enter the elements");
            for(i=0;i<rows;i++)    //Reading
            {
                        for(j=0;j<columns;j++)
                        {
                            scanf("%d",&a[i][j]);
                        }
            }
   printf("elements are\n");
  for(i=0;i<rows;i++)          //displaying
            {
                        for(j=0;j<columns;j++)
                        {
                                    printf("%d\t",a[i][j]);
                         }
                        printf("\n");
            }
}
```

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

```
#include<stdio.h>
int main()
```

```
{
  int m,n,A[20][20],i,j;
  printf("Enter the order of matrix");
  scanf("%d%d",&m,&n);
  printf("Enter the element");
  for(i=0;i<m;i++)
  {
    for(j=0;j<n;j++)
    {
      scanf("%d",&A[i][j]);
    }
  }
printf("Transpose of Matrix A\n");
 for(i=0;i<n;i++)
 {
                for(j=0;j<m;j++)
               {
                        printf("%d\t",A[j][i]);
                }
                 printf("\n");
  }
}
```

Write a program to add the content of two matrices.

```
#include<stdio.h>
int main()
{
  int r1,c1,r2,c2,A[20][20],B[20][20], C[20][20],i,j;
  printf("Enter the order of matrices:");
  scanf("%d%d%d%d",&r1,&c1,&r2,&c2);
  if(r1 != r2 || c1 != c2)
  {
    printf(“Matrices Addition Not possible");
  }
else
{
      printf("Enter the Matrix A\n");
      for(i=0;i<r1;i++)
      {
      for(j=0;j<c1;j++)
      {
        scanf("%d",&A[i][j]);
      }
      }
      printf("Enter the Matrix B\n");
      for(i=0;i<r2;i++)
      {
      for(j=0;j<c2;j++)
      {
         scanf("%d",&B[i][j]);
      }
      }
printf("Added Matrix C:\n");
for(i=0;i<r1;i++)
{
   for(j=0;j<c1;j++)
   {
       C[i][j]=A[i][j] + B[i][j];
      printf("%d\t",C[i][j]);
   }
   printf("\n");
  }
}
}
```

Matrix Multiplication:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \text{ and } B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix}$$

Multiplication result is

$$\begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} & a_{11}b_{13} + a_{12}b_{23} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} & a_{21}b_{13} + a_{22}b_{23} \\ a_{31}b_{11} + a_{32}b_{21} & a_{31}b_{12} + a_{32}b_{22} & a_{31}b_{13} + a_{32}b_{23} \end{bmatrix}$$

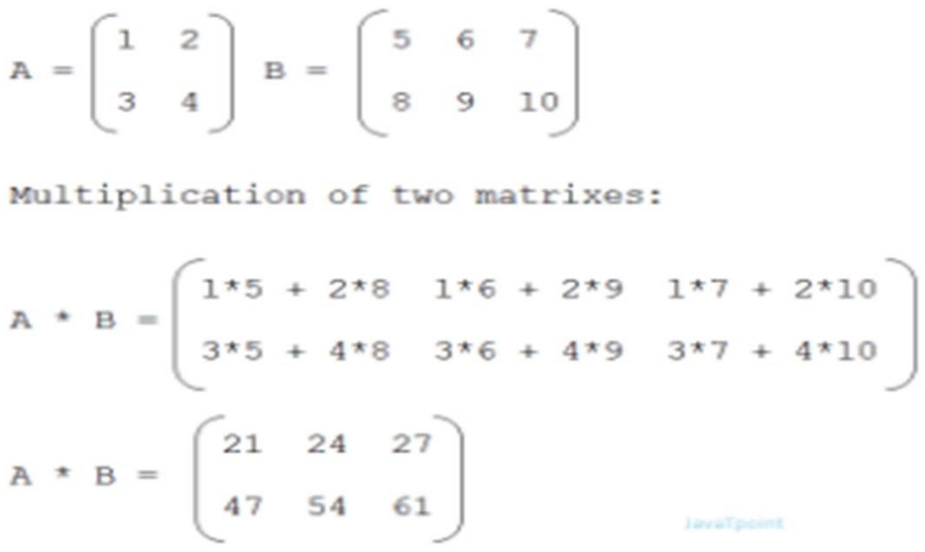
Size of A is r1xc1

Size of B is r2xc2

Size of C is r1xc2

Number of multiplication needed to find single element in C is c1 or r2

Example:



$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 5 & 6 & 7 \\ 8 & 9 & 10 \end{pmatrix}$$

Multiplication of two matrixes:

$$A * B = \begin{pmatrix} 1*5 + 2*8 & 1*6 + 2*9 & 1*7 + 2*10 \\ 3*5 + 4*8 & 3*6 + 4*9 & 3*7 + 4*10 \end{pmatrix}$$

$$A * B = \begin{pmatrix} 21 & 24 & 27 \\ 47 & 54 & 61 \end{pmatrix}$$

Write a program to multiply the content of two matrices.

```
#include<stdio.h>
int main()
{
  int r1,c1,r2,c2,A[20][20],B[20][20],C[20][20],i,j,k;
  printf("Enter the order of matrices:");
  scanf("%d%d%d%d",&r1,&c1,&r2,&c2);
  if(c1 != r2)
  {
    printf("Matrices Multiplication Not possible");
  }
  else
{
            printf("Enter the Matrix A\n");
            for(i=0;i<r1;i++)
            {
            for(j=0;j<c1;j++)
            {
```

```
                        scanf("%d",&A[i][j]);
                    }
            }
            printf("Enter the Matrix B\n");
             for(i=0;i<r2;i++)
           {
                    for(j=0;j<c2;j++)
                    {
                            scanf("%d",&B[i][j]);
                    }
           }
printf("Matrix C:\n");
                    for(i=0;i<r1;i++)
                    {
                            for(j=0;j<c2;j++)
                            {
                                    C[i][j]= 0;
                                    for(k=0;k<c1;k++)
                                    {
                                    C[i][j] += A[i][k] * B[k][j];
                                    }
                                     printf("%d\t",C[i][j]);
                            }
                            printf("\n");
                     }
           }
}
```

STRINGS

Strings are defined as an array of characters. String is a 1D character array.The string is terminated with a special character ‘\0’.

Declaration of strings:

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

```
Eg: char city [10];
    char name[30];
```

When the compiler assigns a character string to a character array, it automatically supplies a null character (\0) at the end of string.Therefore

Size of a string = maximum number of characters in string + one

Initialization can be in either of the following two forms.

```
char city [9] =”NEW YORK”;
char city[9]={‘N’,’E’,’W’,’ ’,’Y’,’O’,’R’,’K’,’\0’};
```

READING STRING FROM KEY BOARD

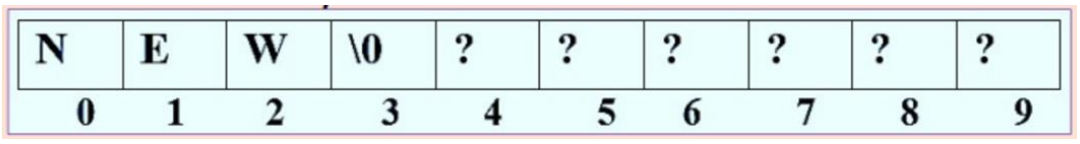
1. Using scanf( ) function

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

In the case of character arrays, the ampersand (&) is not required before the variable name.

The problem with scanf ( ) function is that it terminates its input on the first white space it finds.

If we typed in at the terminal NEW YORK then only the string “NEW” will be read in to the array address. The address array is created in the memory as shown below.



N	E	W	\0	?	?	?	?	?	?
0	1	2	3	4	5	6	7	8	9

scanf ( ) is not capable of receiving multi-word strings. Therefore names such as ‘Hello World’ would be unacceptable.

```
#include <stdio.h>

int main()

{

   char str[20];

   printf("Enter the String\n");

   scanf("%s", str);

   printf("String: %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 1: use % [^\n] instead of %s.

```
#include <stdio.h>

void main()

{

   char str[20];
```

```
    printf("enter the string\n");
    scanf("%[^\n]", str);
    printf("String: %s\n", str);
}
```

Solution 2 : using the function gets ( ).

It does not skip white space.

The usage of functions gets( ) and its counterpart puts( ) is shown below.

```
#include <stdio.h>
int main ( )
{
            char name[25] ;
            printf ( "Enter  full name\n " ) ;
            gets ( name ) ;
            puts ( "Hello!" ) ;
            puts ( name ) ;
}
```

OUTPUT

```
Enter full name
Harun Shan
Hello!
Harun Shan
```

STRING HANDLING FUNCTIONS

String conversion functions are stored in the header file <string.h>

1. strlen - Finds length of a string
2. strlwr - Converts a string to lowercase
3. strupr - Converts a string to uppercase
4. strcat   - Appends one string at the end of another
5. strcpy   - Copies a string into another

6. strcmp - Compares two strings

7. strdup - Duplicates a string

8. strrev - Reverses string

1) strlen ( )

This function counts the number of characters present in a string.

Syntax: strlen(str)

```
#include<stdio.h>
#include<string.h>
int main()
{
  char str[]="Hello World";
  int len;
        len=strlen(str);
    printf("Value=%d",len);
}
```

Output

```
Value=11
```

2. strcat( )

It concatenates two strings and returns the concatenated string.

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

```
#include <stdio.h>
#include <string.h>
int main()
{
     char s1[20] = "Hello";
     char s2[10] = "World";
     strcat(s1,s2);
     printf("After concatenation: %s", s1);
}
```

Output:

```
After concatenation: HelloWorld
```

3. strcpy( )

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

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

Output:

```
String s1 is: Hello World
```

4. strcmp( )

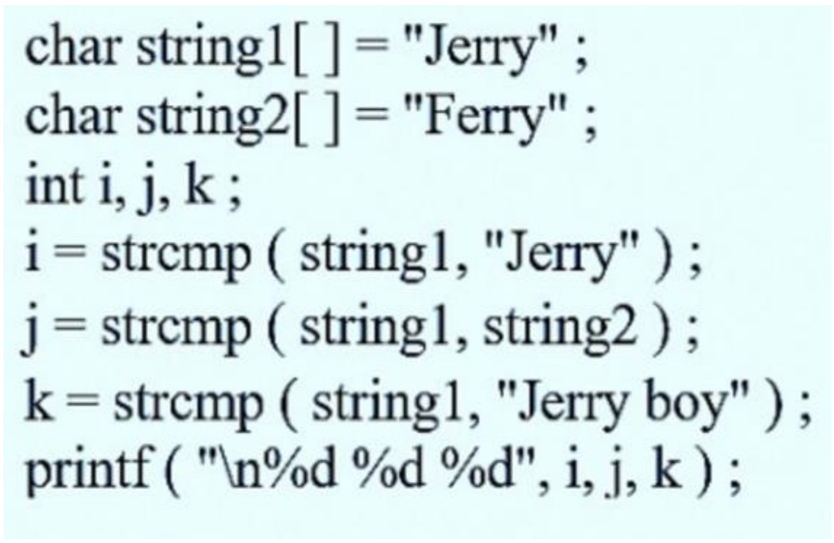
It compares the two strings and returns an integer value. If both the strings are same (equal) then this function would return 0 otherwise it may return a negative or positive value based on the comparison.

Syntax : strcmp(str1,str2)

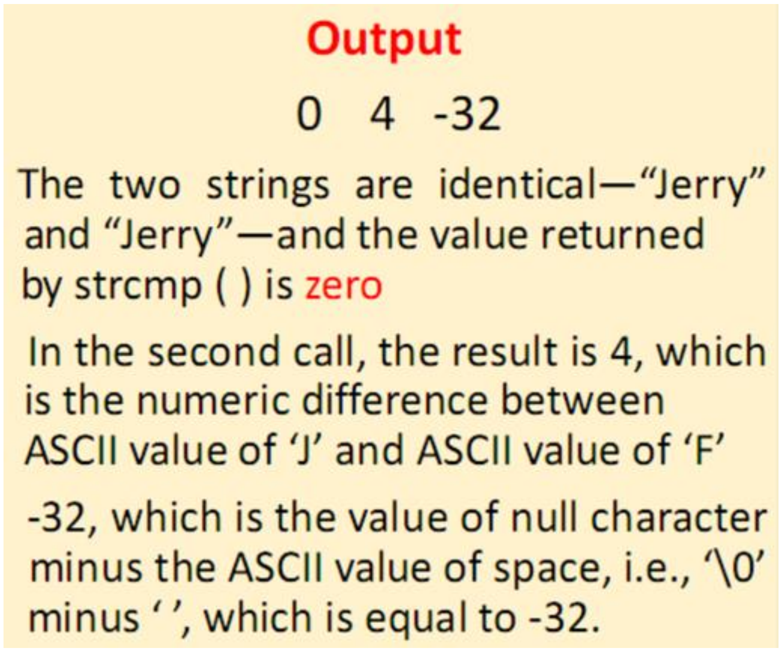
If str1 < str2 then it would result in a negative value.

If str1 > str2 then it would return positive value.

If str1 == str2 then you would get 0(zero)



```
char string1[ ] = "Jerry" ;
char string2[ ] = "Ferry" ;
int i, j, k ;
i = strcmp ( string1, "Jerry" ) ;
j = strcmp ( string1, string2 ) ;
k = strcmp ( string1, "Jerry boy" ) ;
printf ( "\n%d %d %d", i, j, k ) ;
```



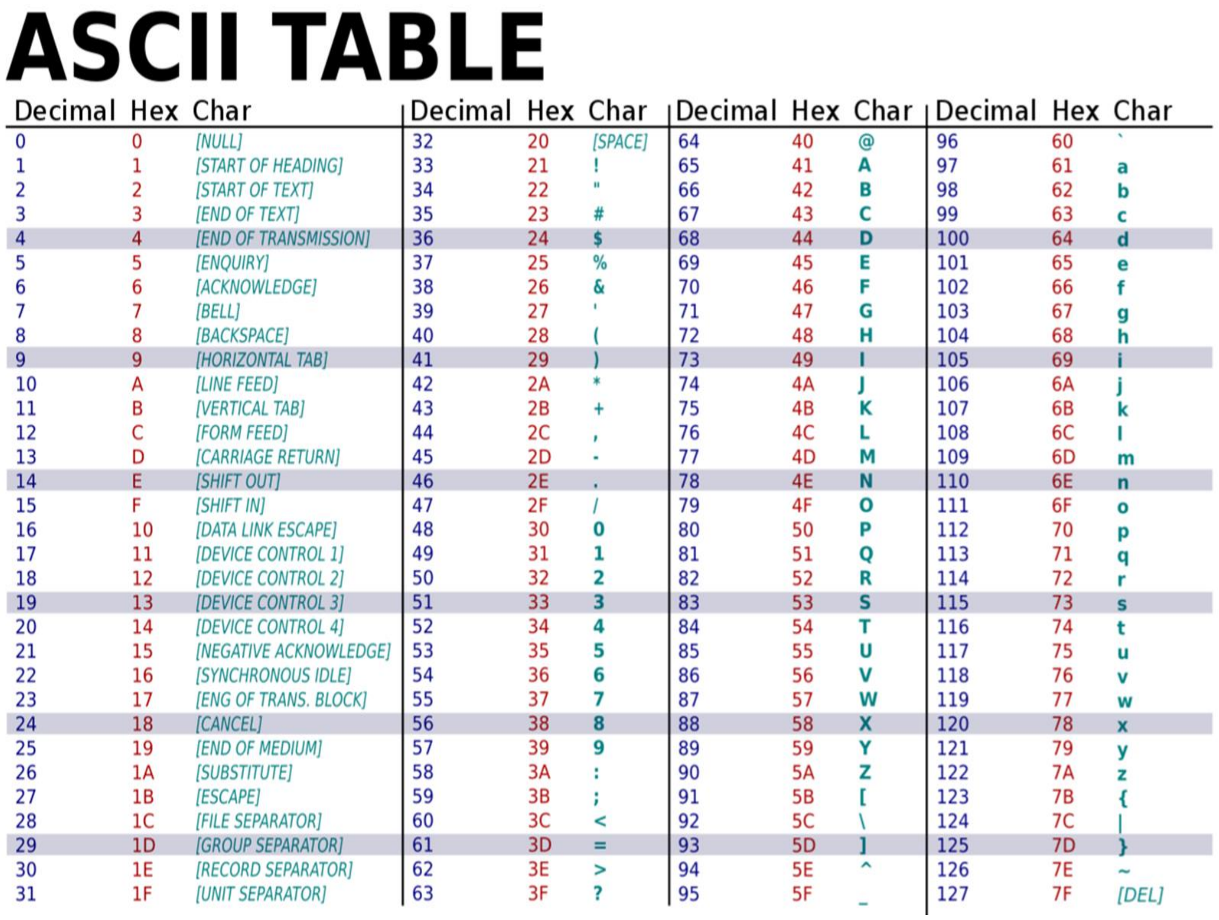
Output

```
0   4   -32
```

The two strings are identical—“Jerry” and “Jerry”—and the value returned by strcmp ( ) is zero

In the second call, the result is 4, which is the numeric difference between ASCII value of ‘J’ and ASCII value of ‘F’

-32, which is the value of null character minus the ASCII value of space, i.e., ‘\0’ minus ‘ ’, which is equal to -32.



ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Program for Checking whether given string is palindrome or not

```
#include <stdio.h>

#include <string.h>

int main()

{

            char one[200] , two[200];

            gets(one);

           strcpy(two, one);

            strrev(two);

            if(strcmp(one, two) == 0)

                        printf("The entered string %s is a palindrome.\n", one);

           else
```

```
                printf("The entered string %s is not a palindrome.\n", one);
        return 0;
}
```