UNIT 4
Syllabus

- ❑ Arrays –Single and Multi-dimensional arrays,
- ❑ Initializing arrays,
- ❑ computing address of an element in array,
- ❑ row major and column major form of an array,
- ❑ character strings and arrays,
- ❑ segmentation fault,
- ❑ bound checking,
- ❑ Sorting Algorithms –

  Bubble sort, insertion sort, selection sort

Arrays (INTRODUCTION)

An array is a collection of similar data elements.

- These data elements have the same data type.
- The elements of the array are stored in consecutive memory locations and are referenced by an index (also known as the subscript).
  - Declaring an array means specifying three things:

data_type array_name[size];

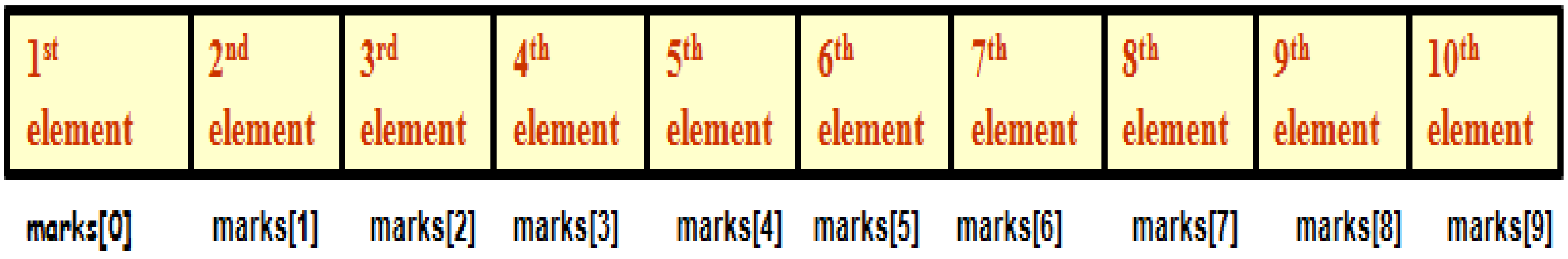
The data type- what kind of values it can store ex, int, char, float

Name- to identify the array

The size- the maximum number of values that the array can hold

For example:

int marks[10];



some examples of array declarations:

```
int age[100];
float sal[15];
char grade[20];
```

Conclusion:

An array is a collection of similar type of data items and each data item is called an element of the array.

The elements of array share the same variable name but each element has a different index number known as subscript.

Arrays can be single dimensional or multidimensional. The number of subscripts determines the dimensions of array. A one-dimensional array has one subscript, two dimensional array has two subscripts.

The one-dimensional arrays are known as vectors and two-dimensional arrays are known as matrices

When the array is declared, the compiler allocates space in memory sufficient to hold all the elements of the array, so the compiler should know the size of array at the compile time.

Hence we can't use variables for specifying the size of array in the declaration.

```
main( )
{
int size=15;
int marks[size];                / *Not valid*/
}
```

Accessing 1-D Array Elements

In C, the array subscripts start from 0. Hence if there is an array of size 5 then the valid subscripts will be from 0 to 4.

The last valid subscript is one less than the size of the array.

This last valid subscript is sometimes know as the upper bound of the array and 0 is known as the lower bound of the array.

Let us take an array:

int arr[5]; /*Size of array arr is 5, can hold five integer elements*/

The elements of this array are-
arr[0], arr[1], arr[2], arr[3], arr[4]

Here 0 is the lower bound and 4 is the upper bound of the array.

In C there is no check on bounds of the array. For example if we have an array arr[5], the valid subscripts are only 0, 1, 2, 3, 4 and if someone tries to access elements beyond these subscripts like arr[5] , arr[10] or arr[-1], the compiler will not show any error message, but this may lead to run time errors, which can be very difficult to debug.

So it is the responsibility of programmer to provide array bounds checking wherever needed.

For example:

```
#include<stdio.h>
main ( )
{
int arr[5],i;
for(i=0;i<3;i++)
{
printf ("Enter the value for arr [%d]",i);
scanf("%d",&arr[i]);

}
for(i=0;i<10;i++)
printf("%d\t",arr[i]);
}
```

We can explicitly initialize arrays at the time of declaration.
The syntax for initialization of an array is:

data_type array_name[size] = {valuel, value2…….valueN };

For example : int marks[5] = {50, 85, 70, 65, 95};

The values of the array elements after this initialization are

marks[0] : 50

marks[1]: 85

marks[2]: 70

marks[3]: 65

marks[4]: 95

If the size is omitted during initialization then the compiler assumes the size of array equal to the number of initializers. For example-

int marks[] = { 99, 78, 50,45, 67, 89};

float sal[] = { 25.5, 38.5, 24.7};

Here the size of array marks is assumed to be 6

and that of sal is assumed to be 3.

- If during initialization the number of initializers is less than the size of array then, all the remaining elements of array are assigned value zero.
- For example-
  int marks[5] = { 99, 78};

  After this initialization the value of the elements are as-
  marks[0] : 99, marks[1]: 78, marks[2]: 0, marks[3]: 0, marks[4]: 0

- So if we initialize an array like this:

  int arr[10] = {0};
  then all the elements of arr will be initialized to zero.
- If the number of initializers is more than the size given in brackets then compiler will show an error.
  For example

  int arr[5] = {1, 2, 3,4,5,6,7, 8};   */*Error*/*

Processing 1-D Arrays

- For processing arrays we generally use a for loop and the loop variable is used at the place of subscript.
- The initial value of loop variable is taken 0 since array subscripts start from zero.
- The loop variable is increased by 1 each time so that we can access and process the next element in the array.
- The total number of passes in the loop will be equal to the number of elements in the array and in each pass we will process one element.

Suppose arr[10] is an array of int type-

(i) Reading values in arr[10]

```
for(i = 0; i < 10; i++)
     scanf("%d", &arr[i]);
```

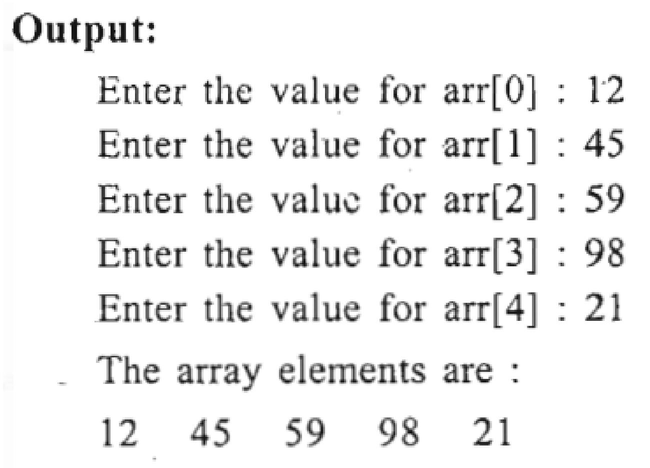
(ii) Displaying values of arr[10]

```
 for(i = 0; i < 10; i++)
     printf(%d ", arr[i]);
```

(iii) Adding all the elements of arr[10]

```
sum = 0;
 for(i = 0; i < 10; i++)
   sum+=arr[i];
```

```
/ * Program to input values into an array and display them* /
#include<stdio.h>
int main( )
{
int arr[5], i;
            for(i=0;i<5;i++)
            {
            printf (“Enter the value for arr[ %d] ”  , i);
            scanf("%d“,&arr[i]) ;
            }

            printf (“The array elements are: \n”);
                        for(i=0;i<5;i++)
                        printf("%d\t",arr[i]);
}
```

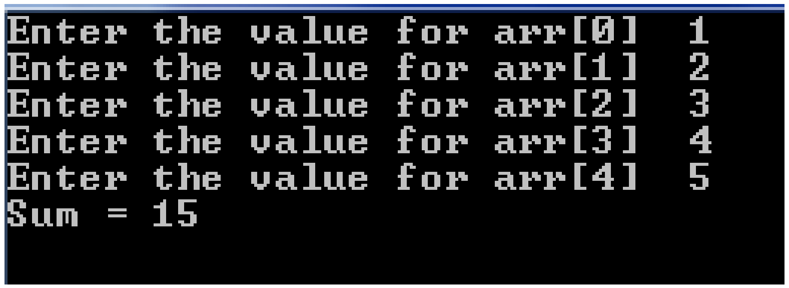


Output:

```
Enter the value for arr[0] : 12
Enter the value for arr[1] : 45
Enter the value for arr[2] : 59
Enter the value for arr[3] : 98
Enter the value for arr[4] : 21
The array elements are :
12    45    59    98    21
```

//Program to add the elements of an array.

```
#include<stdio.h>
main ( )
{
int arr[5],i,sum=0;
    for(i=0;i<5;i++)
    {
    printf ("Enter the value for arr[%d]  ",i);
    scanf("%d",&arr[i]);
    sum+=arr[i];
    }

    printf ("Sum = %d\n", sum) ;
}
```



```
Enter the value for arr[0]  1
Enter the value for arr[1]  2
Enter the value for arr[2]  3
Enter the value for arr[3]  4
Enter the value for arr[4]  5
Sum = 15
```

Program to find the maximum and minimum number in an array

```
#include<stdio.h>
int main( )
{
int i, j , arr[ 10] = {2,5,4,1,8,9,11,6,3,7};
int min, max;
min=max=arr[0];
    for(i=1;i<10;i++)
    {
    if(arr[i]<min)
    min=arr[i];

    if(arr[i]>max)
    max=arr[i];
    }
printf("Minimum = %d, Maximum = %d \n", min, max);
return 0;
}
```

Output:

```
Minimum = 1, Maximum = 11
```

Two Dimensional Array

Declaration and Accessing Individual Elements of a 2-D array:

The syntax of declaration of a 2-D array is similar to that of 1-D arrays, but here we have two subscripts.

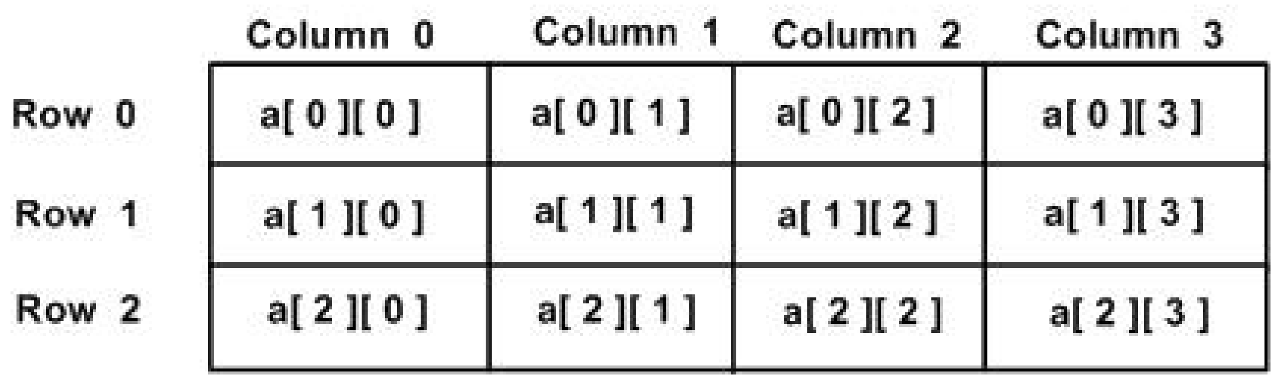
data_type array_name[rowsize][columnsize]; ,
Here row size specifies the number of rows and column size represents the number of columns in the
array.

The total number of elements in the array are, rowsize * columnsize.

For exampleint arr[4][5];
The starting element of this array is arr[0][0] and the last element is arr[3][4]
The total number of elements in this array is 4*5 = 20.

A two-dimensional array can be considered as a table.

For Ex: A two-dimensional array **a**, which contains three rows and four columns can be shown as follows −



	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]

Thus, every element in the array **a** is identified by an element name of the form **a[i][j]**, where 'a' is the name of the array, and 'i' and 'j' are the subscripts that uniquely identify each element in 'a'.

Processing 2-D Arrays

For processing 2-D arrays, we use two nested for loops. The outer for loop corresponds to the row and the inner for loop. corresponds to the column.

For Ex: int arr[4][5];

(i) Reading values in arr

```
for( i = 0;  i < 4;  i++ )
        for( j = 0;  j< 5;  j++ )
                scanf("%d" , &arr[i][j]);
```

(ii) Displaying values of arr

```
for( i = 0;  i < 4;  i++ )
          for( j = 0;  j< 5;  j++ )
                    printf("%d " , arr[i][j]);
```

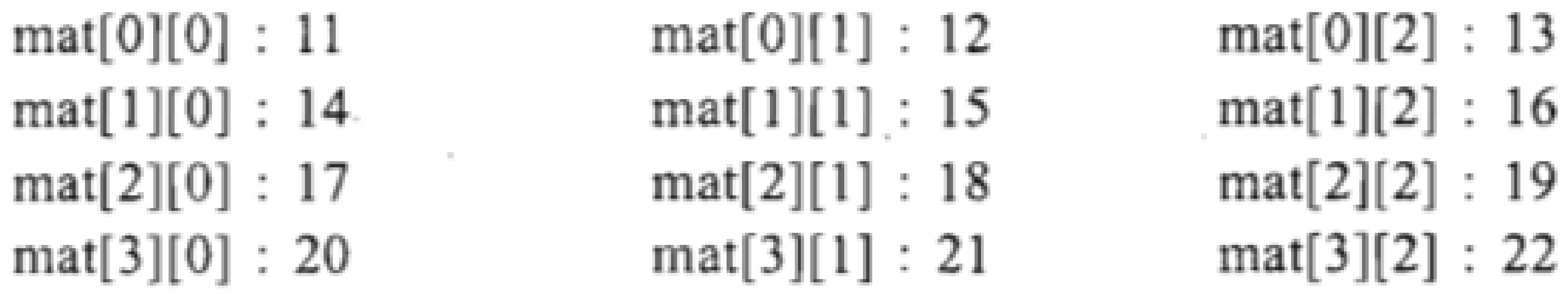
This will print all the elements in the same line. If we want to print the elements of different rows on different lines then we can write like this-

```
for( i = 0;  i < 4;  i++ )
    {
          for( j = 0;  j< 5;  j++ )
                    printf("%d " , arr[i][j]);
                    printf("\n") ;
    }
```

2-D arrays can be initialized in a way similar to that of 1-D arrays. For example

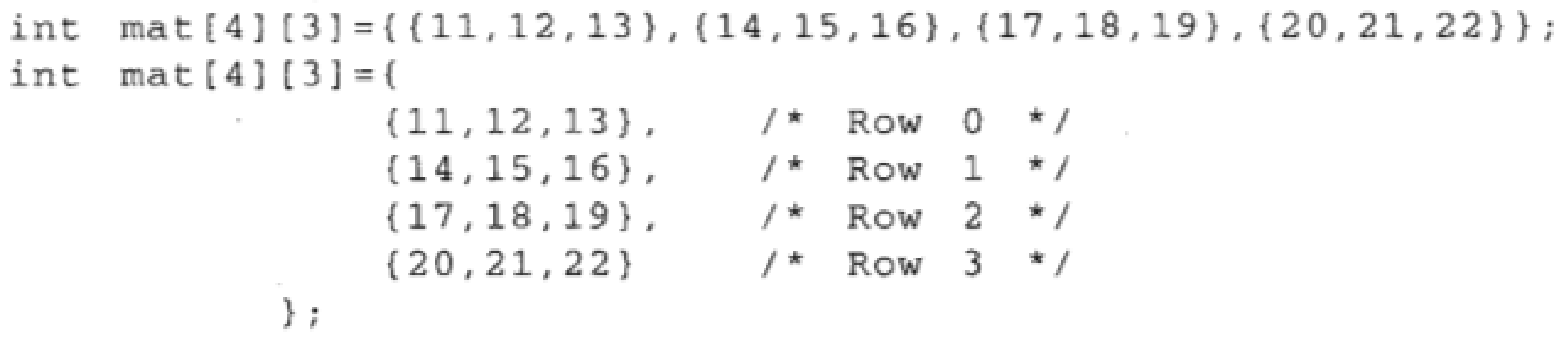
```
int mat[4][3] = { 11, 12, 13, 14, 15, 16, p, 18, 19, 20, 21, 22};
```

These values are assigned to the elements row-wise, so the values of elements after this initialization are:



mat[0][0] : 11	mat[0][1] : 12	mat[0][2] : 13
mat[1][0] : 14	mat[1][1] : 15	mat[1][2] : 16
mat[2][0] : 17	mat[2][1] : 18	mat[2][2] : 19
mat[3][0] : 20	mat[3][1] : 21	mat[3][2] : 22

While initializing we can group the elements row-wise using inner braces. For example



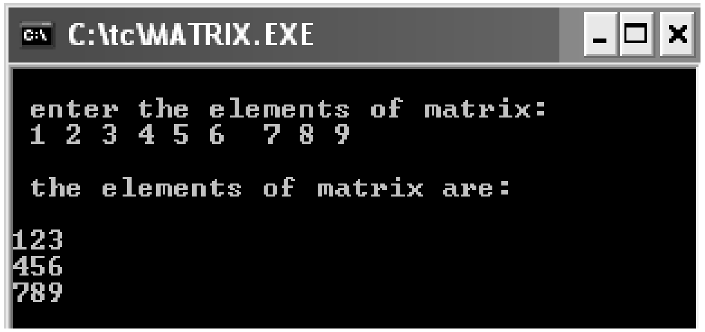
```
int  mat[4][3]={{11,12,13},{14,15,16},{17,18,19},{20,21,22}};
int  mat[4][3]={
                {11,12,13},    /*  Row  0  */
                {14,15,16},    /*  Row  1  */
                {17,18,19},    /*  Row  2  */
                {20,21,22}     /*  Row  3  */
            };
```

used a nested loop to handle a two-dimensional array −

```
#include <stdio.h>
 int main ()
{
     int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};        /* an array with 5 rows and 2
     columns*/
     int i, j;

     for ( i = 0; i < 5; i++ )                                  /* output each array element's value
     */
     {
                         for ( j = 0; j < 2; j++ )
               {
              printf("a[%d][%d] = %d\n", i,j, a[i][j] );
               }
      }

   return 0;
}
```

/* Write a program to read and display a 3 X 3 matrix.*/

```
#include<stdio.h>
void main()
{
 int i, j, arr[3][3];
 printf("\n enter the elements of matrix:\n ");
      for(i=0; i<3; i++)
      {
             for(j=0; j<3; j++)
             {
             scanf("%d", &arr[i][j]);
             }
      }
      printf("\n the elements of matrix are: \n");
      for(i=0; i<3; i++)
      {
      printf("\n");
             for(j=0; j<3; j++)
                        {
             printf("%d", arr[i][j]);
             }
      }


    }
```

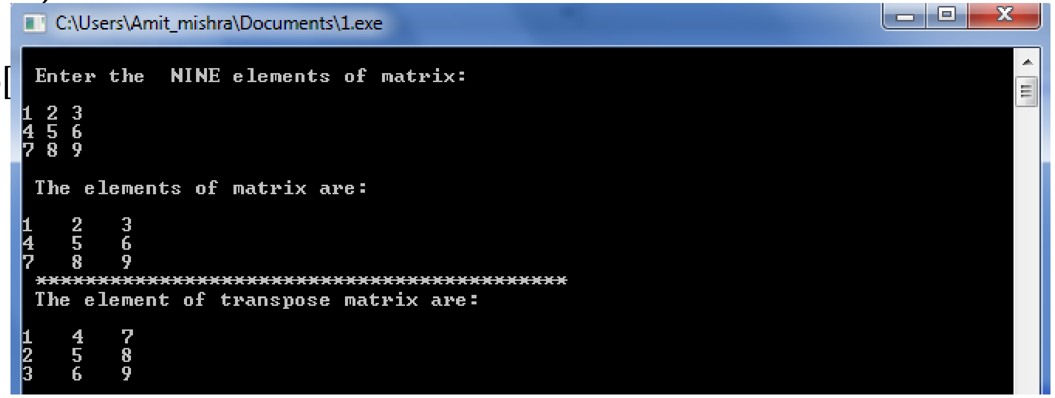


```
C:\tc\MATRIX.EXE

 enter the elements of matrix:
 1 2 3 4 5 6  7 8 9

 the elements of matrix are:

123
456
789
```

/* write a program to transpose a 3 X 3 matrix.*/

```
#include<stdio.h>
int main()
{
 int i, j, a[3][3], b[3][3];\

    printf("\n Enter the  NINE elements of matrix:\n ");
    for(i=0; i<3; i++)
    {
        for(j=0; j<3; j++)
        {
        scanf("%d", &a[i][j]);
        }
    }
    printf("\n The elements of matrix are: \n");
         for(i=0; i<3; i++)
         {
         printf("\n");
                 for(j=0; j<3; j++)
                 {
                 printf("%d", a[i][j]);
                 }
         }
```

```
printf("\n *********************************************");
    for(i=0; i<3; i++)
    {
        for(j=0; j<3; j++)
        {
        b[j][i] = a[i][j];
        }
    }
    printf("\n The element of transpose matrix are: \n");

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



C:\Users\Amit_mishra\Documents\1.exe

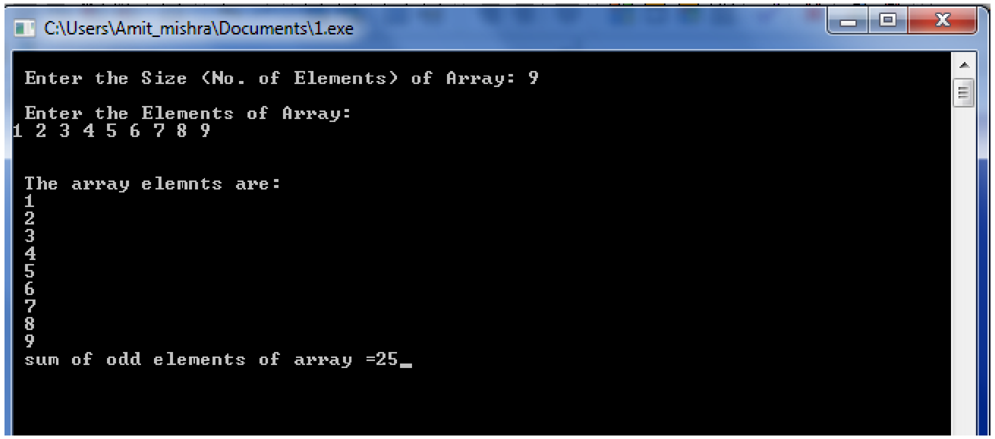
```
 Enter the  NINE elements of matrix:

1 2 3
4 5 6
7 8 9

 The elements of matrix are:

1   2   3
4   5   6
7   8   9
 *******************************************
 The element of transpose matrix are:

1   4   7
2   5   8
3   6   9
```

// Write a program for finding addition of two matrices.

```
#include<stdio.h>
int main()
{
int a[2][2],b[2][2],c[2][2];
int i,j;


printf("enter elements in first matrix\n");
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("enter elements in second matrix\n");
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
scanf("%d",&b[i][j]);
}
}
printf("sum of two matrix is\n");
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
c[i][j]=a[i][j]+b[i][j];
printf("%d   ",c[i][j]);
}
printf("\n");
}

}
```

// write a program for finding the sum of odd elements in an array.

```
#include<stdio.h>

int main()
{
int arr[30];
int n, i, sum=0;

printf("\n Enter the Size (No. of Elements) of Array: ");
scanf("%d", &n);
printf("\n Enter the Elements of Array:\n");
    for(i=0;  i<n; i++)
    {
    scanf("%d", &arr[i]);
    }
printf("\n\n The array elemnts are: ");
    for(i=0; i<n; i++)
{
    printf("\n %d", arr[i]);
}

for(i=0; i<n; i++)
    {
        if(arr[i]%2 != 0)
        sum =sum+arr[i];
    }
printf("\n sum of odd elements of array =%d", sum);
}
```



```
Enter the Size (No. of Elements) of Array: 9

Enter the Elements of Array:
1 2 3 4 5 6 7 8 9


The array elemnts are:
1
2
3
4
5
6
7
8
9
sum of odd elements of array =25_
```

// write a program for copying elements of one array into another array.

```
#include<stdio.h>
#include<conio.h>
int main()
{
int a[5], b[5];
int i;

printf("\n Enter Five Elements in first array:\n");

        for(i=0; i<5; i++)
        {
        scanf("%d", &a[i]);
        }

        for(i=0; i<5; i++)
        {
                b[i] = a[i];
        }
printf("\n After copying, Elements in 2nd array are:\n");
        for(i=0;  i<5;  i++)
        {
        printf("\t  %d", b[i]);
        }
getch();
}
```

// Write a program for finding the number and location in array.

```
#include<stdio.h>
int main()
{
 int arr[5],i,n;
 printf("enter five elements\n");
        for(i=0;i<5;i++)
       {
       scanf("%d",&arr[i]);
       }
 printf("enter element that u want to search\n");
 scanf("%d",&n);
       for(i=0;i<5;i++)
       {
                  if(n==arr[i])
                  {
                  printf("lacation =%d and number=%d",i,arr[i]);
                  printf("adderess =%u",&arr[i]);
                  }
       }
}
```

Introduction To Strings

- In C, strings are treated as arrays of type char and are terminated by a null character('\0').
- This null character has ASCII value zero.
- There is no separate data type for strings in C. They are treated as arrays of type char.
- A character array is a string if it ends with a null character ('\0').
- Strings are generally used to store and manipulate data in text form like words or sentences.

initialization of a string variable:

- char str[10] = {'I', 'n', 'd', 'i', 'a', '\0' };
- char str[10] = "India"; **/*Here the null character is automatically placed at the end*/**

WAP to enter name of your university Name and print it. Consider array

Char Ch[50] ;

```
#include<stdio.h>
main ( )
{
char ch[50];
int i,n;
printf("Enter no. of charcters: \n");
scanf("%d",&n);

printf("Enter the name of your university\n");
    for(i=0;i<n;i++)
    scanf("%c",&ch[i]);

    printf("The name of the university is:\n");

    for(i=0;i<n;i++)
    printf("%c",ch[i]);
}
```

Input and output of strings

```
// Program for input and output of strings using scanf() and printf().
#include<stdio.h>
int main()
{
char str[10]=“GehuDehradun";
printf("String is= %s \t",str);

printf("\n Enter new value for string: ");
scanf ("%s",str) ;
printf ("New String is= %s\n", str);
}
```

OUTPUT:

```
String is= GehuDehradun
 Enter new value for string:  GraphicEra
New String is=  GraphicEra
```

string constant is a sequence of characters enclosed in double quotes. It is sometimes called a literal.
The double quotes are not a part of the string. Some examples of string constants are-

"V"

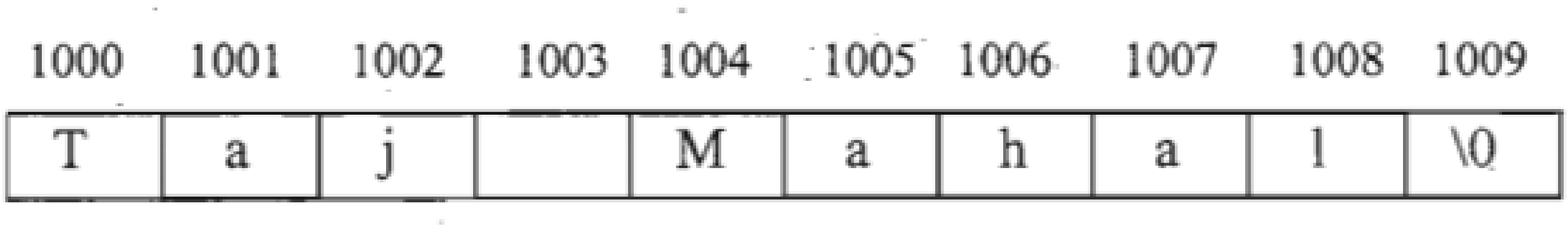
"Taj MahaI"

"2345"

"Subhash Chandra Bose was a great leader"

"My age is %d and height is %f\n" (Control string used in printf)

For example the string "Taj MahaI" will be stored in memory as-



• Each character occupies one byte and compiler automatically inserts the null character at the end

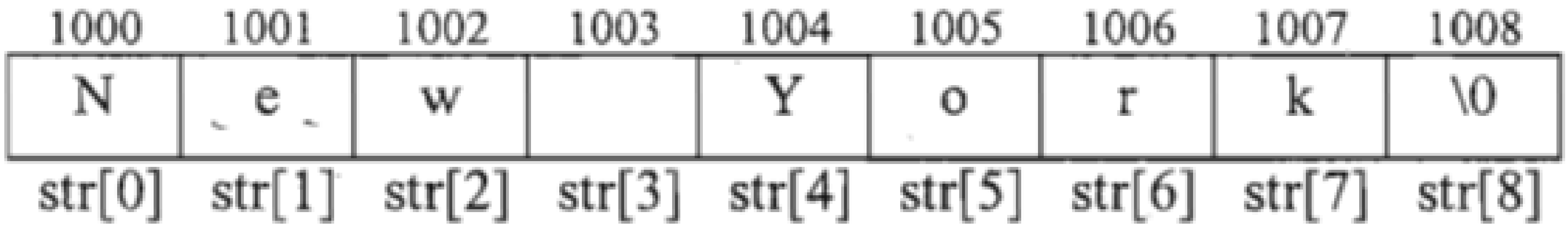
String Variables

To creat a string variable we need to declare a character array with sufficient size to hold all the characters of the string including null character.

char str[ ] = {'N', 'e' , 'w' , ‘ ’ , 'Y‘ , ‘o', 'r’, 'k' ,'\0'}·,

also initialize it as char

str[ ] = "New York";



1000	1001	1002	1003	1004	1005	1006	1007	1008
N	e	w		Y	o	r	k	\0
str[0]	str[1]	str[2]	str[3]	str[4]	str[5]	str[6]	str[7]	str[8]

/*Program to input and output a string variable using scanf( ) */

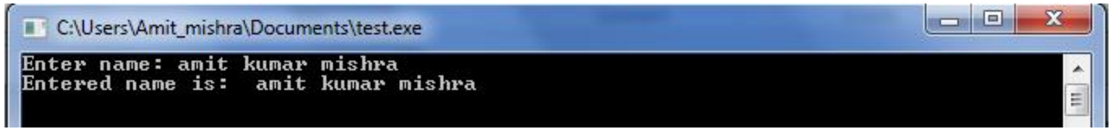
```
#include<stdio.h>
main( )
{
char name[20] ;
printf("Enter name :");
scanf("%s",name) ;
printf("%s ",name);
}
```

gets( ) and puts( )

- scanf( ) stops reading as soon as it encounters a whitespace.
- So for entering string with whitespaces we can use the function **gets( ).**
- It stops reading only when it encounters a newline and replaces this newline by the null character.
- We have another function **puts( )** which can output a string and replaces the null character by a newline.

/*Program to understand the use of gets() and puts( )*/

```
#include<stdio.h>
int main()
{
char name[20];
printf("Enter name: ");
gets(name) ;
printf ("Entered name is:  ");
puts(name);
getch();
return 0;
}
```



```
Enter name: amit kumar mishra
Entered name is:  amit kumar mishra
```

Till Now We have Discussed

- A string is a null-terminated character array.
- This means that after the last character, a null character (**'\0'**) is stored to signify the end of the character array.
- The general form of declaring a string is

```
char str[size];
```

- For example if we write,

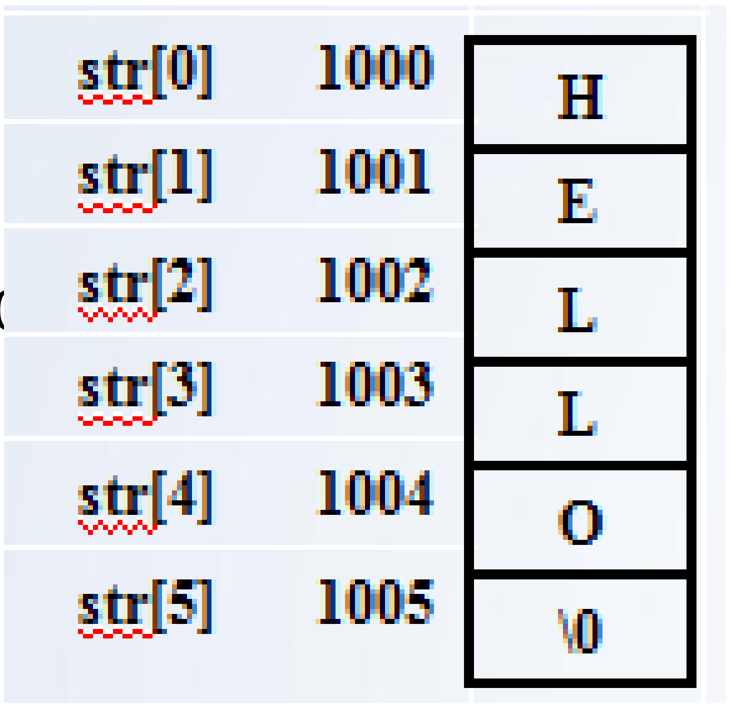
  char str[] = "HELLO";

We are declaring a character array with 5 characters namely, H, E, L, L,

- Besides, a null character ('\0') is stored at the end of the string.
- So, the internal representation of the string becomes: HELLO'\0'

Note that to store a string of length **5**, we need **5 + 1** locations (1 extra for the null character).

- The name of the character array (or the string) is a pointer to the beginning of the string.



str[0]	1000	H
str[1]	1001	E
str[2]	1002	L
str[3]	1003	L
str[4]	1004	O
str[5]	1005	\0

READING STRINGS

For Example:

If we declare a string by writing

char str[100];

Then str can be read from the user by using three ways

use **scanf()** function

using **gets()** function

using **getchar()** function repeatedly

- The string can be read using scanf() by writing

  scanf(“%s”, str);

- The string can be read by writing

  gets(str);

  gets() takes the starting address of the string which will hold the input. The string inputted using gets() is automatically terminated with a null character.

- The string can also be read by calling the **getchar()** repeatedly to read a sequence of single characters (unless a terminating character is entered) and simultaneously storing it in a character array.

WRITING STRINGS

The string can be displayed on screen using three ways

- use **printf()** function
- using **puts()** function
- using **putchar()** function repeatedly

The string can be displayed using printf() by writing

printf("%s", str);

The string can be displayed by writing

puts(str);

The string can also be written by calling the **putchar()** repeatedly to print a sequence of single characters

String Library Functions

- There are several library functions used to manipulate strings.
- The prototypes for these functions are in header file **string.h**.

#include< string.h >

strlen( )

- This function returns the length of the string i.e. the number of characters in the string excluding the terminating null character.
- It accepts a single argument, which is pointer to the first character of the string.

- For example strlen("suresh") returns the value **6**.
- Similarly if s1 is an array that contains the name "deepali" then strlen(s1) returns the value **7** .

Program to understand the work of strlen () function

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

printf ("Enter the string: \n") ;
scanf ("%s", str);

length=strlen(str) ;

printf("Length of the string is  %d\n", length);
}
```

strcmp( )

- This function is used for comparison of two strings.
- If the two strings match, strcmp() returns value 0,
- otherwise it returns a non-zero value.
- This function compares the strings character by character.
- The non-zero value returned on mismatch is the difference of the ASCII value of the non-matching characters of the two strings.

strcmp(s1, s2) returns a value-

< 0 when s1 < s2

= 0 when s1 = = s2

> 0 when s1 > s2

Program to understand the work of strcmp() function

```
#include<stdio.h>
#include<string.h>
int main( )
{
char str1[10], str2[10];

    printf ("Enter the first string ") ;
    scanf("%s",str1);

    printf ("Enter the second string :");
    scanf("%s",str2);
        if( (strcmp(str1,str2) )==0)
        printf ("Stririgs are same\n");
                else
                printf("Strings are not same\n");
}
```

strcpy( )

- This function is used for copying one string to another string.

  strcpy(str1, str2) copies str2 to str1**.**
- Here str2 is the source string and str1 is destination string.
- If str2 = "suresh" then this function copies
  “suresh” into str1.
- *This function takes pointers to two strings as arguments and returns the pointer to first string.*

Program to understand the work of strcpy () function

```
#include<stdio.h>
#include<string.h>
int main( )
{
char str1[10], str2[10];
printf ("Enter the first string: " );
scanf("%s",str1);
printf ("Enter tne second string: " ) ;
scanf("%s",str2) ;
strcpy(str1,str2);
printf ("First string : %s \t\t Second string : %s\n",str1,str2);
strcpy(str1,"Delhi");
strcpy(str2,"Calcutta");
printf ("First string : %s \t\t Second string : %s\n",str1,str2);
}
```

strcat( )

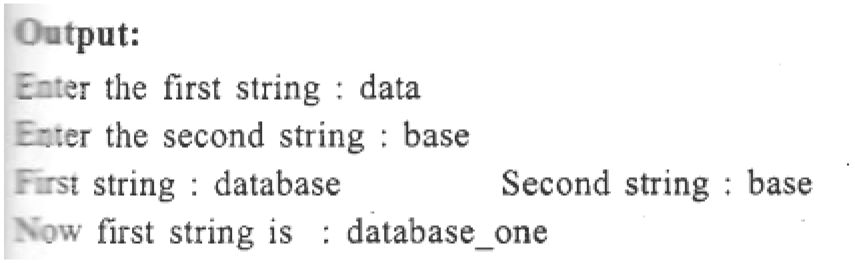
- This function is used for concatenation of two strings.
- If first string is “Graphic" and second string is “Era” then after using this function the first string becomes “GraphicEra".

strcat(str1, str2); **/*concatenates str2 at the end of str1 */**

- The null character from the first string is removed, and the second, string is added at the end of first string. The second string remains unaffected.
- This function takes pointer to two strings as arguments and returns a pointer to the first(concatenated) string.

Program to understand the work of strcat( ) function.

```
#include<stdio.h>
#include<string.h>
int main( )
{
char str1[20],str2[20];
printf ("Enter the first string: ");
scanf("%s",str1);
printf ("Enter the second string: ");
scanf("%s",str2) ;
strcat(str1,str2);
printf( "First string %s \t Second string %s \n", str1, str2);
strcat(str1, "_one");
printf ("Now first string is:  %s \n", str1) ;
}
```



```
Output:
Enter the first string : data
Enter the second string : base
First string : database          Second string : base
Now first string is  : database_one
```