Unit 7: Array

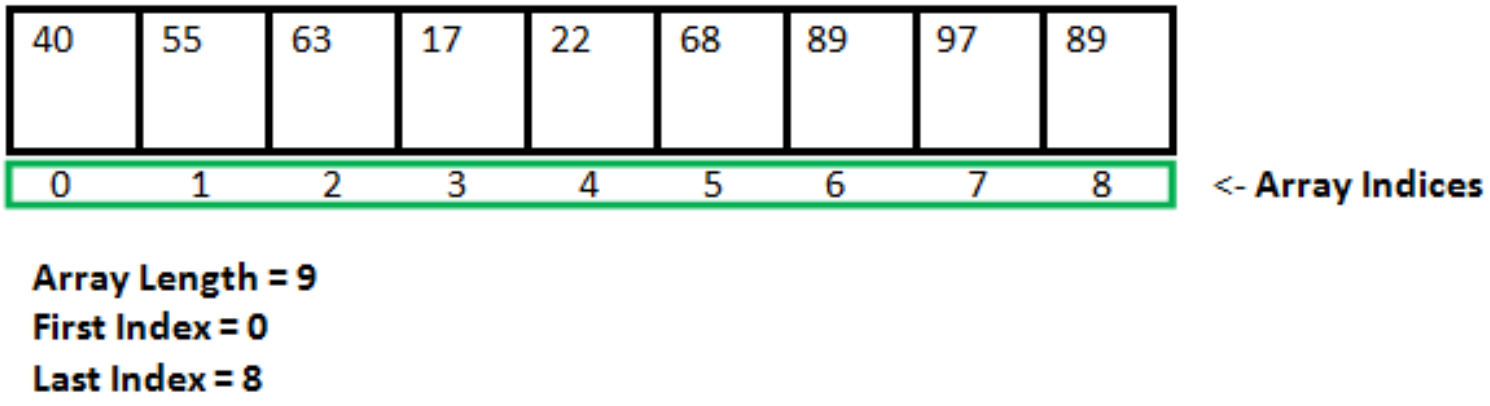
Er.Nipun Thapa

Array

- An array in C or be it in any programming language is a collection of similar data items stored at contiguous memory locations and elements can be accessed randomly using indices of an array.
- They can be used to store collection of primitive data types such as int, float, double, char, etc of any particular type.
- To add to it, an array in C can store derived data types such as the structures, pointers etc.

Array

- Given below is the picturesque representation of an array.

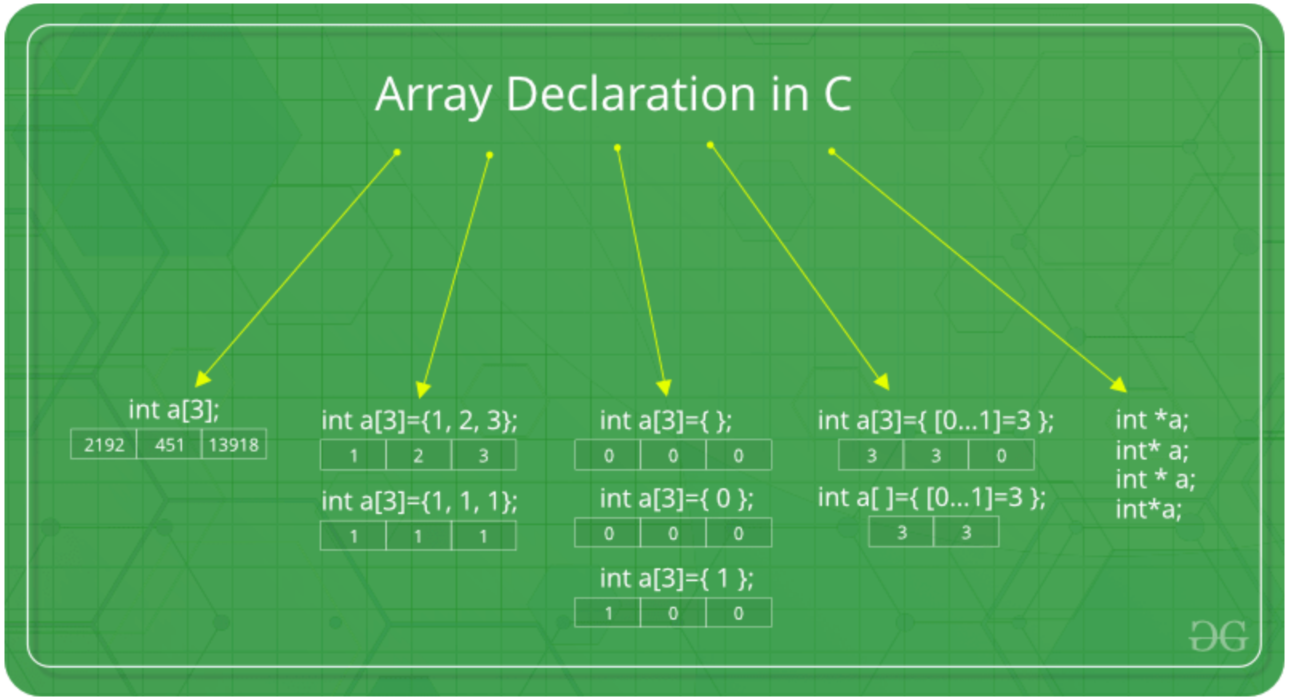


Why do we need arrays?

- 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.

Array declaration in C:

data_type array_name[array_size];



Array declaration by specifying size

```
// Array declaration by specifying size
    int arr1[10];

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

// Array declaration by initializing elements
   int arr[] = { 10, 20, 30, 40 }

// Compiler creates an array of size 4.
// above is same as  "int arr[4] = {10, 20, 30, 40}"
```

Advantages of an Array in C:

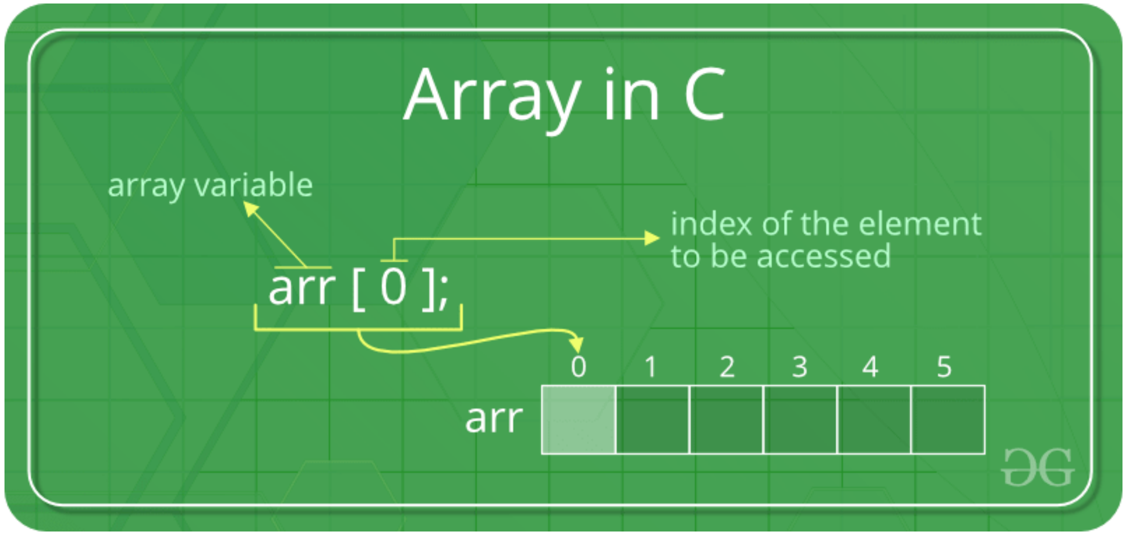
1. Random access of elements using array index.
2. Use of less line of code as it creates a single array of multiple elements.
3. Easy access to all the elements.
4. Traversal through the array becomes easy using a single loop.
5. Sorting becomes easy as it can be accomplished by writing less line of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

Disadvantages of an Array in C:

1. Allows a fixed number of elements to be entered which is decided at the time of declaration. Unlike a linked list, an array in C is not dynamic.
2. Insertion and deletion of elements can be costly since the elements are needed to be managed in accordance with the new memory allocation.

Facts about Array in C:

- **Accessing Array Elements:**
  Array elements are accessed by using an integer index. Array index starts with 0 and goes till size of array minus 1.
- Name of the array is also a pointer to the first element of array.



LAB 5: QN 1: Display the 5 number using array

```
int main()
{
    int i , a[5];
    printf(“Enter No.:”);
    for(i=0;i<5;i++)
    {
            printf(“n[%d]=“,i);
            scanf(“%d”,&a[i]);
    }
    for(i=0;i<5;i++)
    {
            printf(“n[%d]=%d“,i,a[i]);
    }
    return 0;
}
```

LAB 5: QN 2: Sum of 5 number using array

```
int main()
{
    int i , a[5],sum=0;
    printf(“Enter No.:”);
    for(i=0;i<5;i++)
    {
            printf(“n[%d]=“,i);
            scanf(“%d”,a[i]);
    }
    for(i=0;i<5;i++)
    {
            sum=sum+a[i];
    }
    printf(“Sum=%d”,sum);
    return 0;
}
```

LAB 5: QN 3: Program to find the smallest and the largest element in the array

```
int main()
{
   int i,n,a[100],small,large;
   printf("Enter value of n:");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      printf("n[%d]:",i);
      scanf("%d",&a[i]);
   }
   small=a[0];
   large=a[0];
for(i=0;i<n;i++)
   {
      if(small>a[i])
         small=a[i];
      if(large<a[i])
         large=a[i];
   }
   printf("Small=%d\nLarge=%d",small,large);
   return 0;
}
```

LAB 5: QN 4: Program to sorting element using array

```
int main()
{
   int i,n,a[100],small,large,j,temp;
   printf("Enter value of n:");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      printf("n[%d]:",i);
      scanf("%d",&a[i]);
   }

   for(i=0;i<n-1;i++)
   {
      for(j=i+1;j<n;j++)
      {
         if(a[i]>a[j])
         {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
         }
      }
   }

 printf("Ascending:\n");
   for(i=0;i<n;i++)
      printf("%d\t",a[i]);
   printf("\nDescending:\n");
   for(i=n-1;i>=0;i--)
      printf("%d\t",a[i]);
   return 0;
}
```

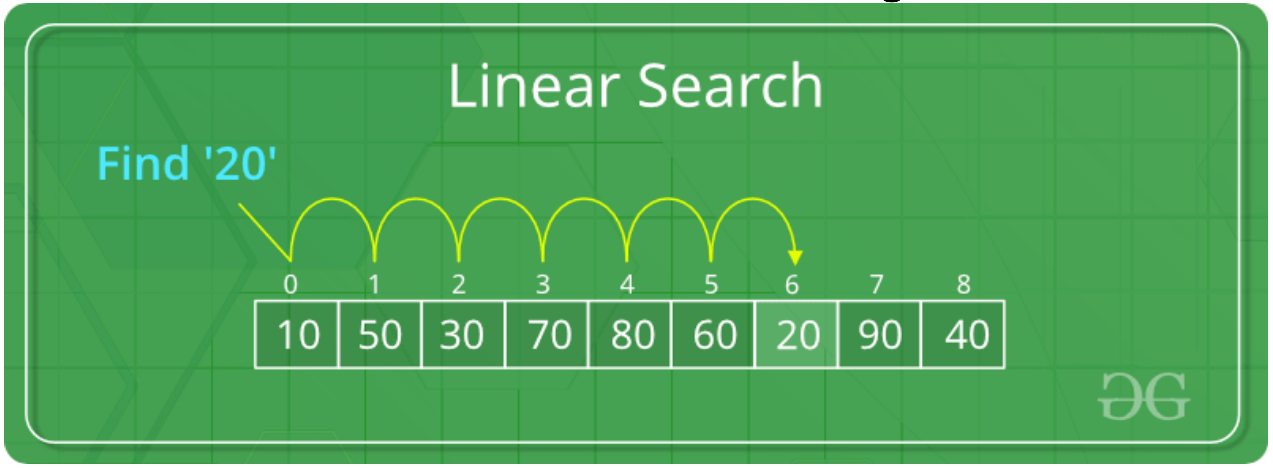
LAB 5: QN 5: Program to sorting element using bubble sort

```
int main()
{
   int i,n,a[100],small,large,j,temp;
   printf("Enter value of n:");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      printf("n[%d]:",i);
      scanf("%d",&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("Ascending:\n");
     for(i=0;i<n;i++)
        printf("%d\t",a[i]);
     printf("\nDescending:\n");
     for(i=n-1;i>=0;i--)
        printf("%d\t",a[i]);
     return 0;
}
```

Searching in an Array

- Searching Algorithms are designed to check for an element or retrieve an element from any data structure where it is stored.
- **Sequential Search**: In this, the list or array is traversed sequentially and every element is checked.

Linear Search to find the element "20" in a given list of numbers



LAB 5: QN 6: Program to searching using array

```
int main()
{
      int arr[]={12,23,78,98,67,56,45,19,65,9},key,i,flag=0;
      printf("\nENTER A NUMBER: ");
      scanf("%d",&key);
      for(i=0;i<10;i++)
      {
                    if(key==arr[i])
                    {
                                flag=1;
                                break;
                    }
      }
      if(flag==1)
                    printf("\nTHE NUMBER %d EXISTS IN THE ARRAY",key);
      else
       printf("\nTHE NUMBER %d DOES NOT EXIST IN THE ARRAY",key);
      return 0;
}
```

Multi-dimensional Arrays in C

- C programming language allows multidimensional arrays. Here is the general form of a multidimensional array declaration −

  type name[size1][size2]...[sizeN];

- For example, the following declaration creates a three dimensional integer array −

  int threedim[5][10][4];

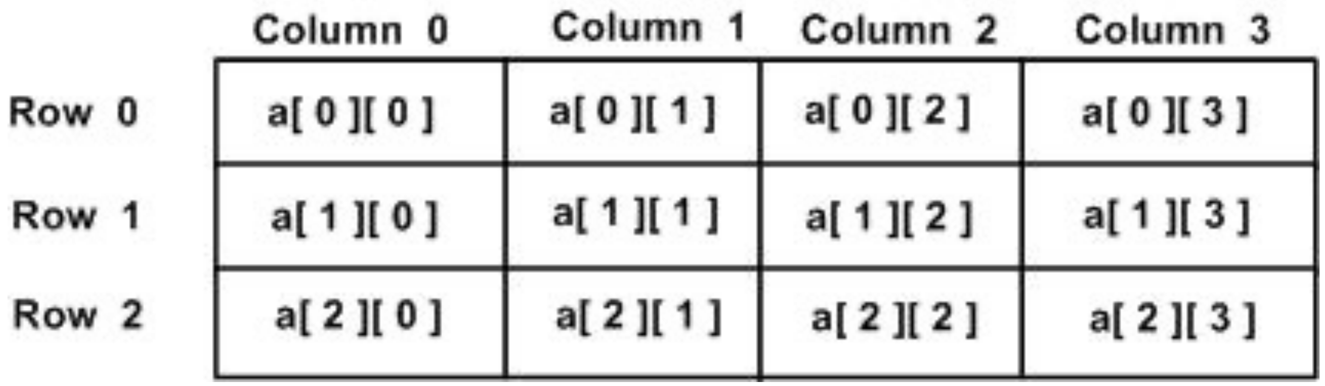
Two-dimensional Arrays

- The simplest form of multidimensional array is the two-dimensional array.
- A two-dimensional array is, in essence, a list of one-dimensional arrays. To declare a two-dimensional integer array of size [x][y], you would write something as follows –

```
type arrayName [ x ][ y ];
```

Two-dimensional Arrays

- Where **type** can be any valid C data type and **arrayName** will be a valid C identifier.
- A two-dimensional array can be considered as a table which will have x number of rows and y number of columns. A two-dimensional array **a**, which contains three rows and four columns can be shown as follows −



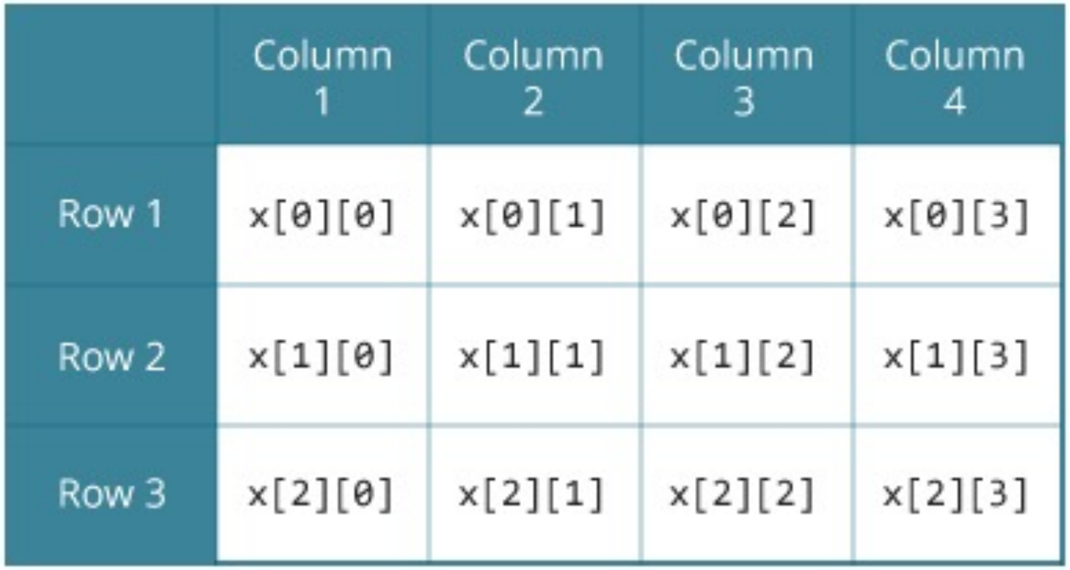
	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

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'.

Two-dimensional Arrays

For example : **float x[3][4];**

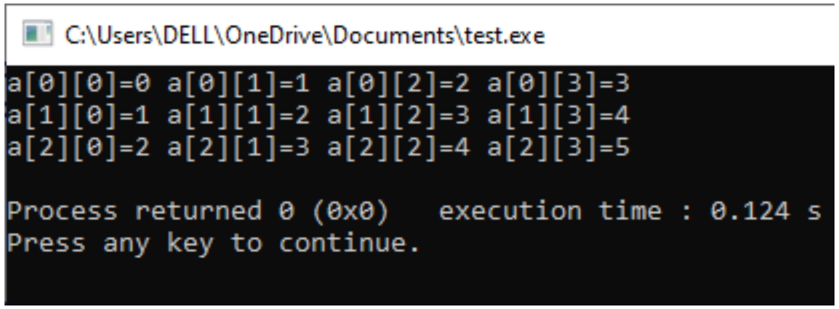
- Here, x is a two-dimensional (2d) array. The array can hold 12 elements. You can think the array as a table with 3 rows and each row has 4 columns.



	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

LAB 5 QN7: Program to demonstrate the loading or storing of data in two dimension array.

```
int main()
{
     int a[3][4],i,j;
     for(i=0;i<3;i++)
    {
       for(j=0;j<4;j++)
           a[i][j]=i+j;
    }
    for(i=0;i<3;i++)
    {
       for(j=0;j<4;j++)
           printf("a[%d][%d]=%d ",i,j,a[i][j]);
       printf("\n");
    }

     return 0;
}
```



C:\Users\DELL\OneDrive\Documents\test.exe

```
a[0][0]=0 a[0][1]=1 a[0][2]=2 a[0][3]=3
a[1][0]=1 a[1][1]=2 a[1][2]=3 a[1][3]=4
a[2][0]=2 a[2][1]=3 a[2][2]=4 a[2][3]=5

Process returned 0 (0x0)   execution time : 0.124 s
Press any key to continue.
```

LAB 5 QN8: Program to read two matrices and display their sum and differences.

```
#include<stdio.h>
int main()
{
   int mat1[3][3],mat2[3][3],sum[3][3],i,j;
   printf("Enter 1st matrix :");
   for(i=0;i<3;i++)
   {
      for(j=0;j<3;j++)
      {
         printf("Mat1[%d][%d]:",i,j);
         scanf("%d",&mat1[i][j]);
      }
   }
   printf("Enter 2nd matrix :");
   for(i=0;i<3;i++)
   {
      for(j=0;j<3;j++)
      {
         printf("Mat2[%d][%d]:",i,j);
         scanf("%d",&mat2[i][j]);
      }
   }
for(i=0;i<3;i++)
   {
      for(j=0;j<3;j++)
      {
         sum[i][j]=mat1[i][j]+mat2[i][j];
      }
   }
   printf("\nThe sum:\n");
   for(i=0;i<3;i++)
   {
      for(j=0;j<3;j++)
        printf("%d\t",sum[i][j]);
      printf("\n");
   }
   return 0;
}
```

LAB 5 QN9: Program to find transpose matrix.

```
#include<stdio.h>
int main()
{
    int mat[3][3],tran[3][3],sum[3][3],i,j;
    printf("Enter 1st matrix :");
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("Mat[%d][%d]:",i,j);
            scanf("%d",&mat[i][j]);
        }
    }
    printf("\nThe matrix to be transposed:\n");
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
          printf("%d\t",mat[i][j]);
        printf("\n");
    }
```

```
 for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            tran[j][i]=mat[i][j];
        }
    }
    printf("\nThe transpose matrix is:\n");
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
          printf("%d\t",tran[i][j]);
        printf("\n");
    }
    return 0;
}
```

LAB 5 QN10: Program to find 3x3 matrix multiplication.

```
#include<stdio.h>
int main()
{
   int mat1[3][3],mat2[3][3],sum[3][3],i,j,k;
   printf("Enter matrix 1:\n");
   for(i=0;i<3;i++)
   {
      for(j=0;j<3;j++)
      {
         printf("Mat1[%d][%d]=",i,j);
         scanf("%d",&mat1[i][j]);
      }
   }
   printf("Enter matrix 2:");
   for(i=0;i<3;i++)
   {
      for(j=0;j<3;j++)
      {
         printf("Mat1[%d][%d]=",i,j);
         scanf("%d",&mat2[i][j]);
      }
   }
printf("\nYour matrix 1 is :\n");
  for(i=0;i<3;i++)
  {
     for(j=0;j<3;j++)
     {
        printf("%d\t",mat1[i][j]);
     }
     printf("\n");
  }
  printf("\nYour matrix 2 is :\n");
  for(i=0;i<3;i++)
  {
     for(j=0;j<3;j++)
     {
        printf("%d\t",mat2[i][j]);
     }
     printf("\n");
  }
//Multiplication calculation
  for(i=0;i<3;i++)
  {
     for(j=0;j<3;j++)
     {
        sum[i][j]=0;
        for(k=0;k<3;k++)
        {
           sum[i][j]=sum[i][j]+mat1[i][k]*mat2[k][j];
        }
     }
  }
  printf("\nMultiplication :\n");
  for(i=0;i<3;i++)
  {
     for(j=0;j<3;j++)
     {
        printf("%d\t",sum[i][j]);
     }
     printf("\n");
  }
  return 0;
}
```

LAB 5 QN11: Program to find the sum of square in a diagonal of a square matrix matrix.

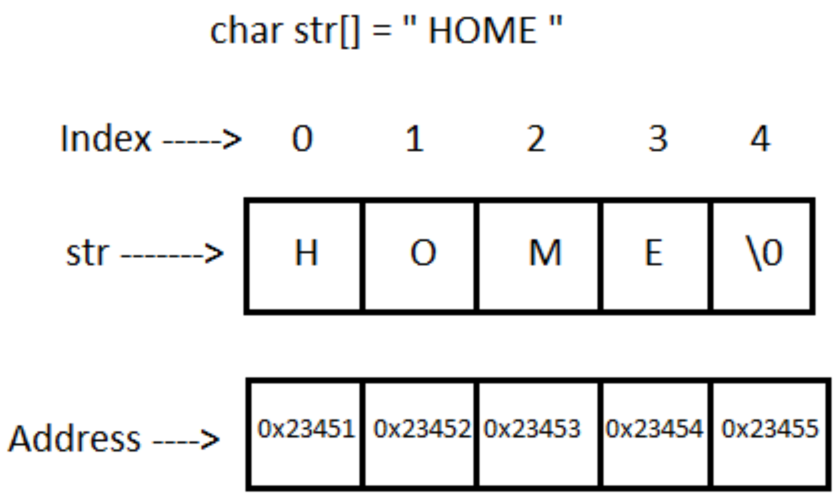
```
#include<stdio.h>
int main()
{
   int mat[3][3],sum=0,i,j;
   printf("Enter matrix :\n");
   for(i=0;i<3;i++)
   {
      for(j=0;j<3;j++)
      {
         printf("Mat[%d][%d]=",i,j);
         scanf("%d",&mat[i][j]);
      }
   }
   printf("\nYour matrix  is :\n");
   for(i=0;i<3;i++)
   {
      for(j=0;j<3;j++)
      {
         printf("%d\t",mat[i][j]);
      }
      printf("\n");
   }
 //calculation
   for(i=0;i<3;i++)
   {
      for(j=0;j<3;j++)
      {
         if(i==j)
            sum=sum+mat[i][j]*mat[i][j];
      }
   }
   printf("\nThe sum of square of element
on a diagonal is : %d",sum);
   return 0;
}
```

Character Array and String

- **String** is a sequence of characters that are treated as a single data item and terminated by a null character '\0’.
- Remember that the C language does not support strings as a data type.
- A **string** is actually a one-dimensional array of characters in C language.
- These are often used to create meaningful and readable programs.

Character Array and String

For example: The string "home" contains 5 characters including the '\0' character which is automatically added by the compiler at the end of the string.



Declaring and Initializing a string variables:

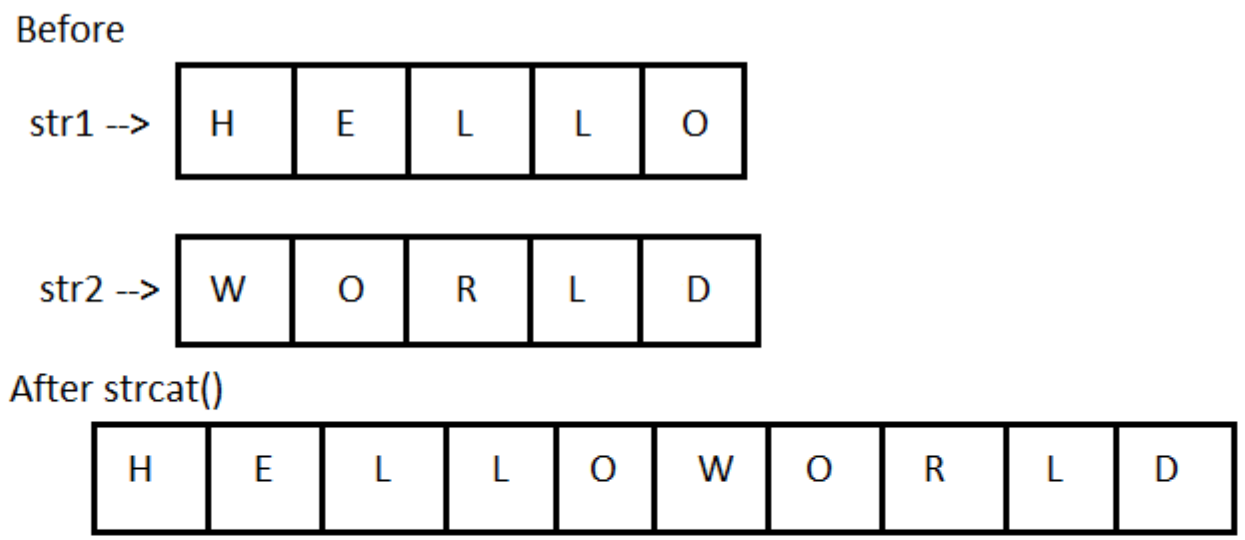
```
// valid
char name[13] = "StudyTonight";
char name[10] = {'c','o','d','e','\0'};

// Illegal
char ch[3] = "hello";
char str[4];
str = "hello";
```

String Handling Functions:

- These functions are packaged in the **string.h** library. Hence, you must include **string.h** header file in your programs to use these functions.
- The following are the most commonly used string handling functions.

Method	**Description**
strcat()	It is used to concatenate(combine) two strings
strlen()	It is used to show the length of a string
strrev()	It is used to show the reverse of a string
strcpy()	Copies one string into another
strcmp()	It is used to compare two string

strcat() function in C:



Syntax:
strcat("hello", "world");Copy
strcat() will add the string **"world"** to **"hello"** i.e ouput = helloworld.

strlen() and strcmp() function:

- strlen() will return the length of the string passed to it and strcmp() will return the ASCII difference between first unmatching character of two strings.

```
        int j = strlen("studytonight");
        int i=strcmp("study ", "tonight");
        printf("%d %d",j,i);
```

```
o/p : 12    -1
```

strcpy() function:

- It copies the second string argument to the first string argument.
- Example of strcpy() function:

```
#include<stdio.h>
#include<string.h>
int main()
{           char s1[50], s2[50];
            strcpy(s1, "StudyTonight");
            strcpy(s2, s1);
            printf("%s\n", s2);
            return 0 ;
}
```

o/p: StudyTonight

strrev() function:

- It is used to reverse the given string expression.

```
#include <stdio.h>
int main()
{
   char s1[50];
   printf("Enter your string: ");
   gets(s1);
   printf("\nYour reverse string is: %s",strrev(s1));
   return(0);
}
```

o/p :

```
Enter your string: studytonight
Your reverse string is: thginotyduts
```

Null character in C

- The Null character in the C programming language is used to terminate the character strings.
- In other words, the Null character is used to represent the end of the string or end of an array or other concepts in C.
- The end of the character string or the **NULL byte** is represented by ‘0’ or ‘\0’ or simply NULL.
- The NULL character doesn’t have any designated symbol associated with it and also it is not required consequently.
- That’s the major reason it is used as a string terminator.

Note: The memory space for each character stored by NULL is 1 byte.

Null Character in C

- Not only strings or arrays, there are various concepts in the C programming language which are terminated by a NULL byte.
- In concepts like Arrays, string literals, character strings a NULL byte is used to indicate the end of the string.
- This can be well illustrated using an example of an array.

Example:

- Suppose there is an array of size 10 and we need to store a string "computer" inside it. This can be simply done using the following piece of code;

```
char a[10] = "computer";
```

When this piece of code is executed, an array of size 10 is created with string "computer" inside it which looks something like this;

c	o	m	p	u	t	e	r	\0	

Here, the '\0' is used to indicate the end of a string.

Finished

Unit 7