2023

DSA
PART-1

CYBER TRON

C++

(THIRD PART)

T.I.M. HAMIM

Index: DSA < Part - 1 >

13.----------

14.----------

15.----------

16.----------

17.----------

18.----------

19.----------

20.----------

21.----------

22.----------

23.----------

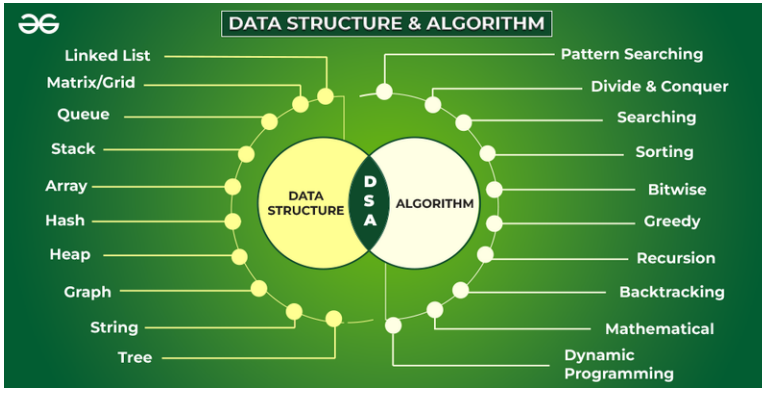
24.----------

25.----------

1. Intro to DSA

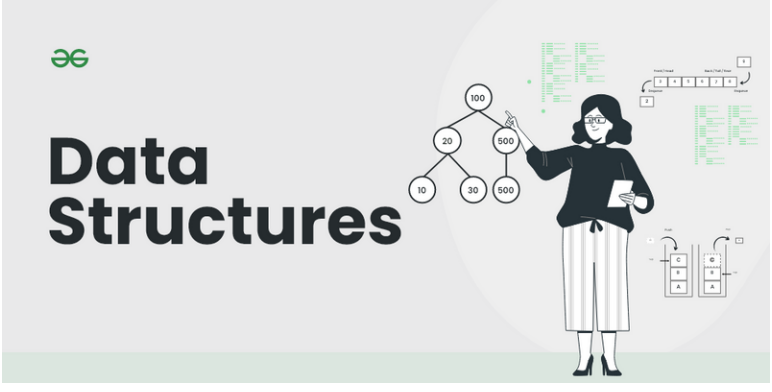
The term DSA stands for Data Structures and Algorithms, in the context of Computer Science.

DSA is defined as a combination of two separate yet interrelated topics – Data Structure and Algorithms. DSA is one of the most important skills that every computer science student must-have. It is often seen that people with good knowledge of these technologies are better programmers than others and thus, crack the interviews of almost every tech giant.

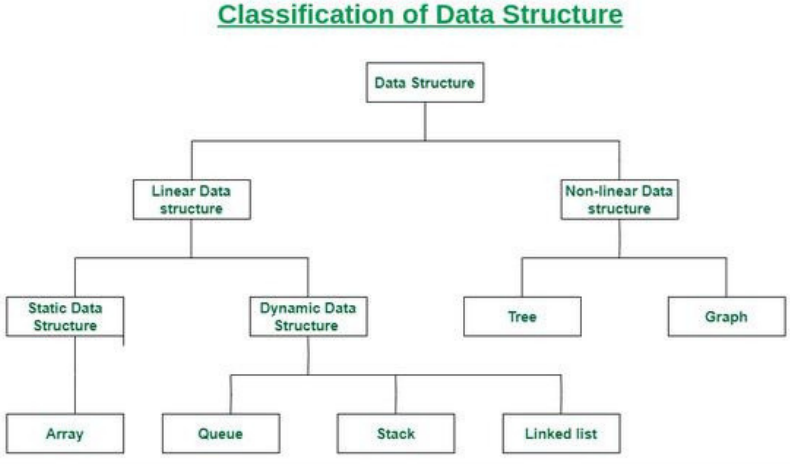


- **What is Data Structure:**

A data structure is a storage that is used to store and organize data. It is a way of arranging data on a computer so that it can be accessed and updated efficiently.



- **Classification of Data Structure:**



(1) Linear data structure: Data structure in which data elements are arranged in one dimension also known as linear dimension, where each element is attached to its previous and next adjacent elements, is called a linear data structure.

Example: array, stack, queue, linked list, etc.

- **Static data structure:** Static data structure has a fixed memory size. It is easier to access the elements in a static data structure.

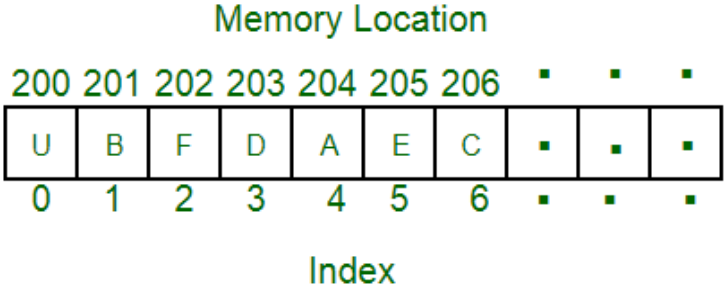
Example: array.

- **Dynamic data structure:** In dynamic data structure, the size is not fixed. It can be randomly updated during the runtime which may be considered efficient concerning the memory (space) complexity of the code.

Example: queue, stack, etc.

(2) Non-linear data structure: Data structures where data elements are not placed sequentially or linearly are called non-linear data structures. In a non-linear data structure, we can't traverse all the elements in a single run only.
Examples of non-linear data structures are **trees** and **graphs.**

For example, we can store a list of items having the same data-type using the array data structure.



- **Applications of Data Structures:**

Data structures are used in various fields such as:

- Operating system
- Graphics
- Computer Design
- Blockchain
- Genetics
- Image Processing
- Simulation,
- etc.

3. Arrays

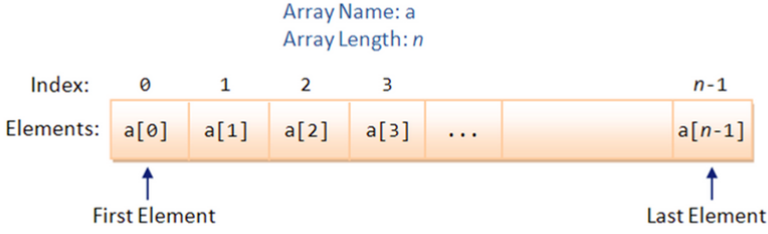
An array is a collection of variables of same data type.

Arrays can following types :

1.One dimensional (1-D) arrays Linear arrays.
Example : int marks[10];
2. Multi dimensinal arrays
(a) Two dimensional (2-D) arrays or Matrix arrays
Example : int marks[2][3];
(b) Three dimensional (3-D) arrays
Example : int marks[2][3][2];

- C++ programming language provides a data structure called the array, which can store a fixed-size sequential collection of elements of the same type.
- An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.
- Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables.
- A specific element in an array is accessed by an index.

- All arrays consist of contiguous memory locations.
- The lowest address corresponds to the first element and the highest address to the last element.



1-D/One dimensional array :

Decliaring Arrays :

- To declare an array in C++, a programmer specifies the type of the elements and the number of elements required by an array as follows :

```
type arrayName [ arraySize ];
```

- This is called a single-dimensional array. The arraySize must be an integer constant greater than zero and type can be any valid C++ data type.
- For example, to declare a 10-element array called balance of type double, use this statement:

```
double balance[10];
```

- Now balance is a variable array which is sufficient to hold up to 10 double numbers.

Initializing Arrays :

- You can initialize array in C++ either one by one or using a single statement as follows:

```
double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

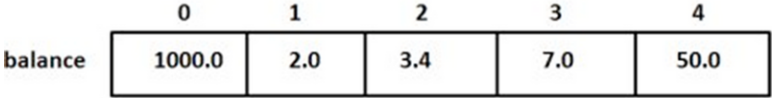
- The number of values between braces { } can not be larger than the number of elements that we declare for the array between square brackets [ ].
- If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write:

```
double balance[] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

- You will create exactly the same array as you did in the previous example.
- Following is an example to assign a single element of the array:

```
balance[4] = 50.0;
```

- The above statement assigns element number 5th in the array with a value of 50.0.
- All arrays have 0 as the index of their first element which is also called base index and last index of an array will be total size of the array minus 1.
- Following is the pictorial representation of the same array we discussed above:



Accessing Array Elements :

- An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array.
- For example:

```
double salary = balance[9];
```

- The above statement will take 10th element from the array and assign the value to salary variable.

Code Example: Array :

- Following is an example which will use all the above mentioned three concepts declaration, assignment and accessing arrays:

```c
#include <stdio.h>

int main ()
{
   int n[ 10 ]; /* n is an array of 10 integers */
   int i,j;

   /* initialize elements of array n to 0 */
   for ( i = 0; i < 10; i++ )
   {
      n[ i ] = i + 100; /* set element at location i to i + 100 */
   }

   /* output each array element's value */
   for (j = 0; j < 10; j++ )
   {
      printf("Element[%d] = %d\n", j, n[j] );
   }

   return 0;
}
```

Output :

```
Element[0] = 100
Element[1] = 101
Element[2] = 102
Element[3] = 103
Element[4] = 104
Element[5] = 105
Element[6] = 106
Element[7] = 107
Element[8] = 108
Element[9] = 109
```

- **Print 10,20,30,40,50 as integer type using array.**

```
#include <iostream>
using namespace std;
int main()
{
 int marks[5];
 marks[0] = 10;
 marks[1] = 20;
 marks[2] = 30;
 marks[3] = 40;
 marks[4] = 50;

 cout<<marks[0] <<" ";
 cout<<marks[1] <<" ";
 cout<<marks[2] <<" ";
 cout<<marks[3] <<" ";
 cout<<marks[4] <<" ";
 return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
 int marks[5];
 marks[0]=10;
 marks[1]=20;
 marks[2]=30;
 marks[3]=40;
 marks[4]=50;
```

```
printf("%d %d %d %d %d",marks[0],marks[1],
 marks[2],marks[3],marks[4]);
 return 0;
}
```

0r,

```
#include <iostream>
using namespace std;
int main()
{
  int marks[5]={10,20,30,40,50};
  printf("%d %d %d %d %d",marks[0],marks[1],
  marks[2],marks[3],marks[4]);
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
 int marks[5]={10,20,30,40,50},i;
 for (i = 0 ; i <=4 ; i++)
 {
  cout<<marks[i]<<" ";
 }
 return 0;
}
```

Output:
10 20 30 40 50

- **What is the output of the following C++ program fragment:**

```
#include <iostream>
using namespace std;
int main()
{
  int marks[5];
  cout<<"marks[0]=";
  cin>>marks[0];
  cout<<"marks[1]=";
  cin>>marks[1];;
  cout<<"marks[2]=";
  cin>>marks[2];;
  cout<<"marks[3]=";
  cin>>marks[3];;
  cout<<"marks[4]=";
  cin>>marks[4];;
  cout<<marks[0]<<" "<<marks[1]<<" "<<
  marks[2]<<" "<<marks[3]<<" "<<marks[4];
  return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
  int marks[5];
  cout<<"marks[0]=";
  scanf("%d",&marks[0]);
  cout<<"marks[1]=";
  scanf("%d",&marks[1]);
  cout<<"marks[2]=";
  scanf("%d",&marks[2]);
  cout<<"marks[3]=";
  scanf("%d",&marks[3]);
  cout<<"marks[4]=";
  scanf("%d",&marks[4]);
  cout<<marks[0]<<" "<<marks[1]<<" "<<
  marks[2]<<" "<<marks[3]<<" "<<marks[4];
  return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
  int marks[5];
  printf("marks[0]=");
  scanf("%d",&marks[0]);
  printf("marks[1]=");
  scanf("%d",&marks[1]);
  printf("marks[2]=");
```

```
  scanf("%d",&marks[2]);
  printf("marks[3]=");
  scanf("%d",&marks[3]);
  printf("marks[4]=");
  scanf("%d",&marks[4]);
  printf("%d %d %d %d %d",marks[0],marks[1],
  marks[2],marks[3],marks[4]);
  return 0;
}
```

Output:

```
marks[0]=10
marks[1]=20
marks[2]=30
marks[3]=40
marks[4]=50
10 20 30 40 50
```

- **What is the output of the following C++ program fragment:**

```
#include <iostream>
using namespace std;
int main()
{
  int marks[5];
  cin>>marks[0];
  cin>>marks[1];
  cin>>marks[2];
  cin>>marks[3];
  cin>>marks[4];
  printf("%d %d %d %d %d",marks[0],marks[1],
  marks[2],marks[3],marks[4]);
  return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
 int marks[5];
 scanf("%d",&marks[0]);
 scanf("%d",&marks[1]);
 scanf("%d",&marks[2]);
 scanf("%d",&marks[3]);
 scanf("%d",&marks[4]);
 printf("%d %d %d %d %d",marks[0],marks[1],
 marks[2],marks[3],marks[4]);
 return 0;
}
```

Output:
10
20
30
40
50
10 20 30 40 50

- **What is the output of the following C++ program fragment:**

```
#include <iostream>
using namespace std;
int main()
{
  int marks[5] = {10, 20, 30, 40, 50}, i;
  for (i = 0; i <= 4; i++)
  {
    cout<<"marks["<<i<<"]"<<"="<<marks[i]<<"\n";
  }
  return 0;
}
```

Output:
marks[0]=10
marks[1]=20
marks[2]=30
marks[3]=40
marks[4]=50

- **What is the output of the following C++ program fragment:**

```
#include <iostream>
using namespace std;
int main()
{
  int marks[5], i;
  for (i = 0; i <= 4; i++)
  {
    scanf("%d", &marks[i]);
  }
  printf("%d %d %d %d %d", marks[0], marks[1],
        marks[2], marks[3], marks[4]);
  return 0;
}
```

Output:

10 20 30 40 50
10 20 30 40 50

- **Do the program given bellow using loop.**

```
#include <stdio.h>
int main()
{
 int marks[5];
 printf("marks[0]=");
 scanf("%d",&marks[0]);
 printf("marks[1]=");
 scanf("%d",&marks[1]);
 printf("marks[2]=");
 scanf("%d",&marks[2]);
 printf("marks[3]=");
 scanf("%d",&marks[3]);
 printf("marks[4]=");
 scanf("%d",&marks[4]);
 return 0;
}
```

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

- **Make a program that will revers the elements of a[ ] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100} as integer type using array.**

```
#include <iostream>
using namespace std;
int main()
{
 int a[] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
 int i, j, b[10];

 for (i = 0, j = 9; i < 10; i++, j--)
 {
  b[j] = a[i];
 }
 for (i = 0; i < 10; i++)
 {
  a[i] = b[i];
 }
 for (i = 0; i < 10; i++)
 {
  cout<<a[i]<<endl;
 }
 return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
 int a[]={10,20,30,40,50,60,70,80,90,100};
 int i,j,temp;

 for(i=0,j=9;i<5;i++,j--){
 temp = a[j];
 a[j] = a[i];
 a[i] = temp;
 }
 for(i=0;i<10;i++){
 cout<<a[i]<<endl;
 }
 return 0;
}
```

- **Make a program that will show sum of two numbers using array.**

```
#include <iostream>
using namespace std;
int main()
{
 int ara[2];
 cout<<"Plase enter two value:\n";
 scanf("%d%d",&ara[0],&ara[1]);
 cout<<"Your result is = "<<ara[0]+ara[1]<<endl;
 return 0;
}
```

- **Specifying the length of an arry using macro is considered to be an excellent practice.**
  For example :

```
#include <iostream>
#define n 10
using namespace std;
int main()
{
 int a[n], i;
 for(i=0; i<n; i++){
   cout<<"Enter the input for index "<<i<<": ";
   cin>>a[i];
 }
 cout<<endl<<"Array elements are as follows:\n";
 for(i=0; i<n; i++){
   cout<<a[i]<<" ";
 }
 return 0;
}
```

- **What if number of elements are lesser than the length specified?**
  int arr[10] = {45, 6, 2, 78, 5, 6};

Here,
The remaining locations of the array are filled by value 0.

```
int arr[10] = {45, 6, 2, 78, 5, 6, 0, 0, 0, 0};
```

- **Some facts about 1D array:**

1. If the number of elements are lesser than the length of the array than the rest of the locations are automatically filled by value 0.
2. Easy way to initialize an array with all zeros is given by:

   int arr[10] = {0}
3. At the time of initialization, never leave these flower brakets {} empty and also never exceed the limit of number of elements specified by the length of an array.

   int arr[5] = {}; ❌

   int arr[5] = {1, 2, 3, 4, 5, 6}; ❌

- **Designated Initializers:**

int arr[10] = {1,0,0,0,0,2,3,0,0,0};
We want:
- 1 in position 0
- 2 in position 5
- 3 in position 6

int arr[10] = {[0] = 1, [5] = 2, [6] =3};

This way of initilization is called designated initialization. And each number in the square brackets is said to be a designator.

. No need to bother about the entries containing zeros.
int a[15] = {1,0,0,0,0,2,0,0,0,0,0,0,0,0,0}
int a[15] = {[0] = 1, [5] = 2};

.No need to bother about the order at all.

```
int a[15] = {[0] = 1, [5] = 2};

int a[15] = {[5] = 2, [0] = 1}
```

Both are same

- Designators could be any non-negative integer.
- Compiler will deduce the length of the array from the largest designator in the list.

```
int a[] = {[5] = 90, [20] = 4, [1] = 45, [49] = 78};
```

Because of this designator, maximum length of this array would be 50.,

Finally, no one can stip you from doing this:

```
int a[] = {1, 7, 5, [5] = 90, 6, [8] = 4};
```

=

```
int a[] ={1, 7, 5, 0, 0, 90, 6, 0, 4};
```

```
#include<stdio.h>
int main()
{
  int a[] = {1, 7, 5, [5] = 90, 6, [8] = 4};
  for(int i=0;i<9;i++){
    printf("%d ",a[i]);
  }
  return 0;
}
```

sizeof operator with arrays:

sizeof(name_of_arr) / sizeof (name_of_arr[0])

sizeof(name_of_arr[0])

int a[] = {1,2,3,4,5,6,7,8,9,10}

sizeof(a[0]);

sizeof(a[0]) = 4 bytes

40 / sizeof(a[0])

= 40 / 4

= 10 → number of elements

size of 1 array element x number of elements = size of whole array.

$$\text{number of elements} = \frac{\text{size of whole array}}{\text{size of 1array element}}$$

- **What is the output of the following C program fragment:**

```
#include <iostream>
#define n 10
using namespace std;
int main()
{
 int a[] = {1,2,3,4,5,6,7,8,9,10};
 cout<<"Number of elements : "<<sizeof
(a)/sizeof(a[0]);
 return 0;
}
```

Output:
Number of elements : 10

Constant arrays in C:

Either one dimensional or multi-dimensional arrays can be made constant by starting the declaration with the keyword const.

For example:

```
#include<stdio.h>
int main()
{
 const int a[] = {1,2,3,4,5,6,7,8,9,10};
 a[1]=45;
 printf("%d",a[1]);
 return 0;
}
```

Here, the output will be : ERROR!

- It assures us that the program will not modify the array which may contain some valuable information.
- It also helps the compiler to catch errors by informing that there is no intention to modify this array.

- **Write a program that can store a constant value of pi.**

```
#include <iostream>
const float pi=3.1416;
using namespace std;
int main()
{
 printf("pi = %.4f",pi);
 return 0;
}
```

- **Make a program that will show sum and average of 10,20,30,40,50 using array.**

```
#include <iostream>
const float pi=3.1416;
using namespace std;
int main()
{
 int marks[5]={10,20,30,40,50},i,sum=0;
 for(i=0;i<=4;i++){
  sum=sum+marks[i];
 }
 cout<<"Sum is : "<<sum<<endl;
 printf("Average is %.2f",(float)sum/5);
 return 0;
}
```

Output:

```
Sum is : 150
Average is 30.00
```

Here,
Type cast : (float)sum/5
Using type cast we make 'sum/5' which into float was in integer type.

2-D/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 as follows:

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

- A two-dimensional array can be think as a table which will have x number of rows and y number of columns.
- A 2-dimensional array a, which contains three rows and four columns can be shown as below:

	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 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.

Initializing Two-Dimensional Arrays:

- Multidimensional arrays may be initialized by specifying bracketed values for each row.
- Following is an array with 3 rows and each row has 4 columns.

```
int a[3][4] = {
 {0, 1, 2, 3} ,   /*  initializers for row indexed by 0 */
 {4, 5, 6, 7} ,   /*  initializers for row indexed by 1 */
 {8, 9, 10, 11}   /*  initializers for row indexed by 2 */
};
```

- The nested braces, which indicate the intended row, are optional. The following initialization is equivalent to previous example:

```
int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

Accessing Two-Dimensional Array Elements ;

An element in 2-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array. For example:

```
int val = a[2][3];
```

- The above statement will take 4th element from the 3rd row of the array. You can verify it in the above diagram.

Example :

Let us check below program where we have used nested loop to handle a two dimensional array:

```
#include <stdio.h>

int main ()
{
   /* an array with 5 rows and 2 columns*/
   int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};
   int i, j;

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

Output :

```
a[0][0]: 0
a[0][1]: 0
a[1][0]: 1
a[1][1]: 2
a[2][0]: 2
a[2][1]: 4
a[3][0]: 3
a[3][1]: 6
a[4][0]: 4
a[4][1]: 8
```

- **Make a program that will show given bellow using 2-D array.**
  1 2 3 4
  5 6 7 8
  9 10 11 12

```
#include <iostream>
using namespace std;
int main()
{
 int a[3][4]={
  {1,2,3,4},
  {5,6,7,8},
  {9,10,11,12}
 };
 printf("%d ",a[0][0]);
 printf("%d ",a[0][1]);
 printf("%d ",a[0][2]);
 printf("%d ",a[0][3]);
 printf("\n");

 printf("%d ",a[1][0]);
 printf("%d ",a[1][1]);
 printf("%d ",a[1][2]);
 printf("%d ",a[1][3]);
 printf("\n");

 printf("%d ",a[2][0]);
 printf("%d ",a[2][1]);
 printf("%d ",a[2][2]);
 printf("%d ",a[2][3]);
 printf("\n");
 return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
 int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}},i,j;
 for(i=0;i<=2;i++){
  for(j=0;j<=3;j++){
   printf("%d ",a[i][j]);
  }
  printf("\n");
 }
 return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
 int n, m, i, j;
 printf("Enter the Raw :");
 scanf("%d", &n);
 printf("Enter the Column :");
 scanf("%d", &m);
 int a[n][m];
 for (i = 0; i < n; i++)
 {
  for (j = 0; j < m; j++)
  {
   printf("a[%d][%d]=",i,j);
   scanf("%d", &a[i][j]);
```

```
  if (j == m - 1)
 {
 printf("\n");
 }
 }
 }
 for (i = 0; i < n; i++)
 {
 for (j = 0; j < m; j++)
 {
 printf("%d ", a[i][j]);
 }
 printf("\n");
 }
 return 0;
}
```

or,

```
#include <iostream>
using namespace std;

int main()
{
 int n, m, i, j;
 printf("Enter the RAw :");
 scanf("%d", &n);
 printf("Enter the Column :");
 scanf("%d", &m);
 int A[n][m];
 //Scaning A matrix
 printf("Enter elements for A matrix :\n");
```

```
for (i = 0; i < n; i++)
 {
 for (j = 0; j < m; j++)
 {

printf("A[%d][%d]=",i,j);
 scanf("%d", &A[i][j]);
 }
 printf("\n");
 }
 //Printing A matrix
 printf("A=");
 for (i = 0; i < n; i++)
 {
 printf("\t");
 for (j = 0; j < m; j++)
 {
 printf("%d ", A[i][j]);
 }
 printf("\n");
 }
 return 0;
}
```

- **Make a program that will show sum of A matrix and B matrix using array.**

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
  int n, m;
  cout << "Enter raw : ";
  cin >> n;
  cout << "Enter column : ";
  cin >> m;
  int arr1[n][m], arr2[n][m], arr3[n][m];

  cout<<endl;

  cout << "Enter elements for 1st matrix : \n"<<endl;

  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < m; j++)
    {
      cout << "arr1[" << i << "]"
         << "[" << j << "] = ";
      cin >> arr1[i][j];
    }
    cout << endl;
  }
```

```
cout<<"1st matrix : \n"<<endl;
cout<<"A = ";
for (int i = 0; i < n; i++)
{
cout<<"\t";
for (int j = 0; j < m; j++)
{
cout << setw(3) << arr1[i][j];
}
cout << "\n\v";
}

cout << "Enter elements for 2nd matrix : \n"<<endl;

for (int i = 0; i < n; i++)
{
for (int j = 0; j < m; j++)
{
cout << "arr2[" << i << "]"
<< "[" << j << "] = ";
cin >> arr2[i][j];
}
cout << endl;
}

cout<<"2nd matrix : \n"<<endl;
cout<<"B = ";
for (int i = 0; i < n; i++)
{
cout<<"\t";
```

```
 for (int j = 0; j < m; j++)
 {
 cout << setw(3) << arr2[i][j];
 }
 cout << "\n\v";
 }

 // Summation of 1st matrix and 2nd matrix
 cout<<"1st matrix + 2nd matrix : "<<endl<<"\n";
 for(int i=0; i<n; i++){
 for(int j=0; j<m; j++){
 arr3[i][j] = arr1[i][j] + arr2[i][j];
 }
 }

 cout<<"A+B = ";
 for(int i=0; i<n; i++){
 cout<<"\t";
 for(int j=0; j<m; j++){
 cout<<setw(3)<<arr3[i][j];
 }
 cout<<"\n\v";
 }
}
```

Output :

Enter raw : 3
Enter column : 4

Enter elements for 1st matrix :

arr1[0][0] = 1
arr1[0][1] = 1
arr1[0][2] = 1
arr1[0][3] = 1

arr1[1][0] = 1
arr1[1][1] = 1
arr1[1][2] = 1
arr1[1][3] = 1

arr1[2][0] = 1
arr1[2][1] = 1
arr1[2][2] = 1
arr1[2][3] = 1

1st matrix :

A = 1 1 1 1

1 1 1 1

1 1 1 1

Enter elements for 2nd matrix :

arr2[0][0] = 2
arr2[0][1] = 2
arr2[0][2] = 2
arr2[0][3] = 2

arr2[1][0] = 3
arr2[1][1] = 3
arr2[1][2] = 3
arr2[1][3] = 3

arr2[2][0] = 4
arr2[2][1] = 4
arr2[2][2] = 4
arr2[2][3] = 4

2nd matrix :

B = 2 2 2 2

3 3 3 3

4 4 4 4

1st matrix + 2nd matrix :

A+B= 3 3 3 3

4 4 4 4

5 5 5 5

- **Make a program that will show sum and subtraction of A and B matrix using array.**

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
  int n, m;
  cout << "Enter raw : ";
  cin >> n;
  cout << "Enter column : ";
  cin >> m;
  int arr1[n][m], arr2[n][m], arr3[n][m];

  cout<<endl;

  cout << "Enter elements for 1st matrix : \n"<<endl;

  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < m; j++)
    {
      cout << "arr1[" << i << "]"
         << "[" << j << "] = ";
      cin >> arr1[i][j];
    }
    cout << endl;
  }
```

```
cout<<"1st matrix : \n"<<endl;
cout<<"A = ";
for (int i = 0; i < n; i++)
{
cout<<"\t";
for (int j = 0; j < m; j++)
{
cout << setw(3) << arr1[i][j];
}
cout << "\n\v";
}

cout << "Enter elements for 2nd matrix : \n"
<<endl;

for (int i = 0; i < n; i++)
{
for (int j = 0; j < m; j++)
{
cout << "arr2[" << i << "]"
<< "[" << j << "] = ";
cin >> arr2[i][j];
}
cout << endl;
}

cout<<"2nd matrix : \n"<<endl;
cout<<"B = ";
for (int i = 0; i < n; i++)
{
cout<<"\t";
```

```
 for (int j = 0; j < m; j++)
 {
 cout << setw(3) << arr2[i][j];
 }
 cout << "\n\v";
 }

 // Subtraction of 1st matrix and 2nd matrix
 cout<<"1st matrix + 2nd matrix : "<<endl<<"\n";
 for(int i=0; i<n; i++){
 for(int j=0; j<m; j++){
 arr3[i][j] = arr1[i][j] - arr2[i][j];
 }
 }

 cout<<"A-B = ";
 for(int i=0; i<n; i++){
 cout<<"\t";
 for(int j=0; j<m; j++){
 cout<<setw(3)<<arr3[i][j];
 }
 cout<<"\n\v";
 }
}
```

- **Make a program that will show multiplication of two matrix A and B using array.**

```
#include<iostream>
#include<iomanip>
using namespace std;

int main(){
  int r1, c1, r2, c2;
  while(true){
    cout<<"Enetr the Raw and Column of A matrix : ";
    cin>>r1>>c1;
    cout<<"Enter the Raw and Column of B matrix : ";
    cin>>r2>>c2;
    if(c1!=r2){
      cout<<"Error!! column of first matrix isd not
equal to raw of second matrix"<<endl;
    }else{
      break;
    }
  }

  int A[r1][c1], B[r2][c2], C[r1][c2];

  // Scaning A matrix
  cout<<"Enter elements for A matrix : "<<endl;
  for(int i=0; i<r1; i++){
    for(int j=0; j<c1; j++){
      cout<<"A["<<i<<"]["<<j<<"] = ";
      cin>>A[i][j];
    }
```

```
cout<<endl;
}

// Scaning B matrix
cout<<"Enter elements for B matrix : "<<endl;
for(int i=0; i<r2; i++){
for(int j=0; j<c2; j++){
cout<<"B["<<i<<"]["<<j<<"] = ";
cin>>B[i][j];
}
cout<<endl;
}

// Printing A matrix
cout<<"A = ";
for(int i=0; i<r1; i++){
cout<<"\t";
for(int j=0; j<c1; j++){
cout<<A[i][j]<<" ";
}
cout<<endl;
}
cout<<endl;

// Printing B matrix
cout<<"B = ";
for(int i=0; i<r2; i++){
cout<<"\t";
for(int j=0; j<c2; j++){
cout<<B[i][j]<<" ";
}
```

```
cout<<endl;
}
cout<<endl;

//Multiplying matrix
int sum=0;
cout<<"A*B = ";
for(int i=0; i<r1; i++){
cout<<"\t";
for(int j=0; j<c2; j++){
for(int k=0; k<c1; k++){
sum = sum + A[i][k]*B[k][j];
}
C[i][j] = sum;
cout<<C[i][j]<<" ";
sum=0;
}
cout<<endl;
}
}
```

Output:

```
Enetr the Raw and Column of A matrix : 3 4
Enter the Raw and Column of B matrix : 4 3
Enter elements for A matrix :
A[0][0] = 1
A[0][1] = 2
A[0][2] = 3
A[0][3] = 4

A[1][0] = 5
A[1][1] = 6
A[1][2] = 7
A[1][3] = 8

A[2][0] = 3
A[2][1] = 6
A[2][2] = 2
A[2][3] = 8

Enter elements for B matrix :
B[0][0] = 4
B[0][1] = 7
B[0][2] = 4

B[1][0] = 3
B[1][1] = 8
B[1][2] = 6

B[2][0] = 3
B[2][1] = 9
B[2][2] = 2
```

B[3][0] = 1
B[3][1] = 7
B[3][2] = 4

A = 1 2 3 4
5 6 7 8
3 6 2 8

B = 4 7 4
3 8 6
3 9 2
1 7 4

A*B = 23 78 38
67 202 102
44 143 84

- **Make a program that will show transpose matrix of A matrix using array.**

```
#include<iostream>
#include<iomanip>
using namespace std;
int main(){
  int r,c;
  cout<<"Enter the Raw of A matrix : ";
  cin>>r;
  cout<<"Enter the Coloumn of A matrix : ";
  cin>>c;
  int A[r][c], T[c][r];

  // Scaning A matrix
  cout<<"Enter elements for A matrix : "<<endl;
  for(int i=0; i<r; i++){
    for(int j=0; j<c; j++){
      printf("A[%d][%d] = ",i,j);
      cin>>A[i][j];
    }
    cout<<endl;
  }

  // Printing A matrix
  cout<<"A = ";
  for(int i=0; i<r; i++){
    cout<<"\t";
    for(int j=0; j<c; j++){
      cout<<setw(3)<<A[i][j];
    }
```

```
 cout<<endl;
 }

 // Transpose matrix
 for(int i=0; i<c; i++){
 for(int j=0; j<r; j++){
 T[i][j] = A[j][i];
 }
 }

 // Printing Transpose matrix
 cout<<"\nT = ";
 for(int i=0; i<c; i++){
 cout<<"\t";
 for(int j=0; j<r; j++){
 cout<<setw(3)<<T[i][j];
 }
 cout<<endl;
 }
}
```

- **Make a program that will show sum of diagonal of A matrix using array.**

```
#include<iostream>
#include<iomanip>
using namespace std;
int main(){
  int r,c;
  cout<<"Enter the Raw of A matrix : ";
  cin>>r;
  cout<<"Enter the Coloumn of A matrix : ";
  cin>>c;
  int A[r][c], sum=0;

  // Scaning A matrix
  cout<<"Enter elements for A matrix : "<<endl;
  for(int i=0; i<r; i++){
    for(int j=0; j<c; j++){
      printf("A[%d][%d] = ",i,j);
      cin>>A[i][j];
    }
    cout<<endl;
  }

  // Printing A matrix
  cout<<"A = ";
  for(int i=0; i<r; i++){
    cout<<"\t";
    for(int j=0; j<c; j++){
      cout<<setw(3)<<A[i][j];
    }
```

```
 cout<<endl;
 }

 //sum of diagonal elements
 cout<<"Diagonal elements : ";
 for(int i=0; i<r; i++){
 for(int j=0; j<c; j++){
 if(i==j){
 cout<<A[i][j]<<" ";
 sum = sum + A[i][j];
 }
 }
 }

 cout<<"\nSum of diagonal matrix : "<<sum<<endl;
}
```

- **Make a program that will show upper and lower triangle elements of A matrix using array.**

```
#include <iostream>
using namespace std;
int main()
{
 int r, c, i, j,uppersum=0,lowersum=0;
 printf("\nEnter the Raw of A matrix :");
 scanf("%d", &r);
 printf("Enter the Column of A matrix :");
 scanf("%d", &c);
 int A[r][c];
 printf("\n");
 //Scaning A matrix
 printf("Enter elements for A matrix :\n\n");
 for (i = 0; i < r; i++)
 {
  for (j = 0; j < c; j++)
  {
   printf("A[%d][%d]=",i,j);
   scanf("%d", &A[i][j]);
  }
  printf("\n");
 }
 //Printing A matrix
 printf("A=");
 for (i = 0; i < r; i++)
 {
  printf("\t");
```

```
 for (j = 0; j < c; j++)
 {
 printf("%d ", A[i][j]);
 }
 printf("\n\n");
 }
 //sum of upper and lower triangle matrix
 for (i = 0; i < r; i++)
 {
 for (j = 0; j < c; j++)
 {
 if(i<=j){
 uppersum=uppersum+A[i][j];
 }
 if(i>=j){
 lowersum=lowersum+A[i][j];
 }
 }
 }
 printf("Sum of upper triangle elements is
%d",uppersum);
 printf("\nSum of lower triangle elements is
%d",lowersum);
 return 0;
}
```

or,

```
#include <iostream>
using namespace std;

const int MAX_SIZE = 100;

void printUpperTriangle(int matrix[MAX_SIZE][MAX_SIZE],
int n) {
    for (int i = 0; i < n; i++) {
        for (int j = i; j < n; j++) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}

void printLowerTriangle(int matrix[MAX_SIZE][MAX_SIZE],
int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j <= i; j++) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}

int main() {
    int n;
    int A[MAX_SIZE][MAX_SIZE];

    cout << "Enter the size of the square matrix (n): ";
    cin >> n;
```

```
    cout << "Enter the elements of the matrix:" << endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> A[i][j];
        }
    }

    cout << "Upper Triangle Elements:" << endl;
    printUpperTriangle(A, n);

    cout << "Lower Triangle Elements:" << endl;
    printLowerTriangle(A, n);

    return 0;
}
```

3-D/Three dimensional array :

A 3D array is a multi-dimensional array(array of arrays). A 3D array is a collection of 2D arrays. It is specified by using three subscripts:Block size, row size and column size. More dimensions in an array means more data can be stored in that array.

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 5 . 10 . 4 integer array:

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

- As explained above, you can have arrays with any number of dimensions, although it is likely that most of the arrays you create will be of one or two dimensions.

Visualizing 3D array:

If we want to visualize a 2D array, we can visualize it in this way:
int arr[3][3], it means a 2D array of type integer having 3 rows and 3 columns.It is just a simple matrix

int arr[3][3]; //2D array contaning 3 rows and 3 columns

1 2 3
4 5 6
7 8 9
3x3

But, what happen if we add one more dimension here,
i.e, int arr[3][3][3], now it becomes a 3D array.

- int shows that the 3D array is an array of type integer.
- arr is the name of array.
- first dimension represents the block size(total number of 2D arrays).
- second dimension represents the rows of 2D arrays.
- third dimension represents the columns of 2D arrays.
- i.e; int arr[3][3][3], so the statement says that we want three such 2D arrays which consists of 3 rows and 3 columns.

int arr[3][3][3]; //3D array

block(1)	block(2)	block(3)
1 2 3	10 11 12	19 20 21
4 5 6	13 14 15	22 23 24
7 8 9	16 17 18	25 25 27
3x3	3x3	3x3

Declaring a 3D array:

To declare 3D array:

- Specify data type, array name, block size, row size and column size.
- Each subscript can be written within its own separate pair of brackets.
- Syntax: data_type array_name[block_size][row_size][column_size];
  Example:

```
int arr[2][3][3];      //array of type integer
                       //number of blocks of 2D arrays:2
                         |rows:3 |columns:3
                       //number of elements:2*3*3=18
```

```
block(1)  11 22 33        block(2)  12 13 14
          44 55 66                  21 31 41
          77 88 99                  12 13 14
                3x3                       3x3
```

Ways to declare 3D array:

1). int arr[2][3][3];
In this type of declaration, we have an array of type integer, block size is 2, row size is 3 and column size is 3.Here we have not stored any values/elements in the array.So the array will hold the garbage values.

int arr[2][3][3]; //no elements are stored

```
 block(1)                block(2)
1221 -543 3421       654 5467 -878     //all values are
3342 6543 4221       456 1567 7890     //garbage values
-564  4566 -345      567 6561 2433
                  3x3                        3x3
```

2). int arr[2][3][3]={};
In this type of declaration, we have an array of type integer, block size is 2, row size is 3 and column size is 3 and we have put curly braces after assignment operator.So the array will hold 0 in each cells of array.

int arr[2][3][3]={}; //0 will be stored

```
block(1)  0 0 0       block(2)  0 0 0
          0 0 0                 0 0 0
          0 0 0                 0 0 0
            3x3                     3x3
```

3). int arr[3][2][2]={0,1,2,3,4,5,6,7,8,9,3,2}
In this type of declaration, we have an array of type integer, block size is 3,
row size is 2, column size is 2 and we have mentioned the values inside the curly braces during the declaration of array. So all the values will be stored one by one in the array cells.

int arr[3][2][2]={0,1,2,3,4,5,6,7,8,9,3,2}

block(1) 0 1 block(2) 4 5 block(3) 8 9
2 3 6 7 3 2
2x2 2x2 2x2

4). int arr[3][3][3]=
{ {{10,20,30},{40,50,60},{70,80,90}},
{{11,22,33},{44,55,66},{77,88,99}},
{{12,23,34},{45,56,67},{78,89,90}}
};
In this type of declaration, we have an array of type integer, block size is 3,
row size is 3, column size is 3 and the values of each blocks are assigned during its declaration.

int arr[3][3][3]=
{ {{10,20,30},{40,50,60},{70,80,90}}, // block 1
{{11,22,33},{44,55,66},{77,88,99}}, //block 2
{{12,23,34},{45,56,67},{78,89,90}} //block 3
};

block(1) block(2) block(3)
10 20 30 11 22 33 12 23 34
40 50 60 44 55 66 45 56 67
70 80 90 77 88 99 78 89 90
3x3 3x3 3x3

Inserting values in 3D array:

In 3D array, if a user want to enter the values then three for loops are used.

- First for loop represents the blocks of a 3D array.
- Second for loop represents the number of rows.
- Third for loop represents the number of columns.

Example:

Following is the implementation in C++:

```
#include <iostream>
using namespace std;
int i,j,k;                    //variables for nested for
loops
int main()
{
 int arr[2][3][3];            //array declaration
 printf("enter the values in the array: \n");
 for(i=1;i<=2;i++)            //represents block
 {
  for(j=1;j<=3;j++)          //represents rows
  {
   for(k=1;k<=3;k++)        //represents columns
   {
    printf("the value at arr[%d][%d][%d]: ",i,j,k);
    scanf("%d",&arr[i][j][k]);
   }
  }
 }
 printf("printing the values in array: \n");
```

```
for(i=1;i<=2;i++)
 {
  for(j=1;j<=3;j++)
  {
   for(k=1;k<=3;k++)
   {
    printf("%d ",arr[i][j][k]);
    if(k==3)
    {
     printf("\n");
    }
   }
  }
  printf("\n");
 }
 return 0;
}
```

Output:

enter the values in the array:

the value at arr[1][1][1]: 23
the value at arr[1][1][2]: 34
the value at arr[1][1][3]: 45
the value at arr[1][2][1]: 76
the value at arr[1][2][2]: 78
the value at arr[1][2][3]: 98
the value at arr[1][3][1]: 87

the value at arr[1][3][2]: 67
the value at arr[1][3][3]: 98
the value at arr[2][1][1]: 34
the value at arr[2][1][2]: 23
the value at arr[2][1][3]: 67
the value at arr[2][2][1]: 58
the value at arr[2][2][2]: 19
the value at arr[2][2][3]: 84
the value at arr[2][3][1]: 39
the value at arr[2][3][2]: 82
the value at arr[2][3][3]: 44

printing the values in array:
23 34 45
76 78 98
87 67 98

34 23 67
58 19 84
39 82 44

In the above program;

- We have declared three variables i,j,k for three for loops.
- we have declared an array of type integer int arr[2][3][3];(blocks:2 rows:3 columns:3)
- First section of nested for loops ask the user to insert the values.
- second section of nested for loops will print the inserted values in the matrix form.

Updating the 3D array:

We can update the elements of 3D array either by specifying the element to be replaced or by specifying the position where replacment has to be done.
For updating the array we require,

- Elements of an array
- Element or position, where it has to be inserted
- The value to be inserted

Example

Following is the implementation in C++ :

```
#include <iostream>
using namespace std;
int i,j,k;                    //variables for nested for loop
int num;                      //will hold the value to be
replaced
int main()
{
 int arr[2][3][3];            //3D array declaration
 printf("enter the values in the array: \n\n");
 for(i=1;i<=2;i++)            //represents block
 {
  for(j=1;j<=3;j++)          //represents rows
  {
   for(k=1;k<=3;k++)    //represents columns
   {
    printf("the value at arr[%d][%d][%d]: ",i,j,k);
    scanf("%d",&arr[i][j][k]);
```

```
 }
 }
}
 printf("\nprinting the values in array: \n");
 for(i=1;i<=2;i++)
 {
  for(j=1;j<=3;j++)
  {
   for(k=1;k<=3;k++)
   {
    printf("%d ",arr[i][j][k]);
    if(k==3)
    {
     printf("\n");
    }
   }
  }
  printf("\n");
 }
 printf("\nenter the block row and column number:
\n");
 scanf("%d %d %d ",&i,&j,&k);         //position where
we want to update the element
 printf("enter the new number you want to update
with: ");
 scanf("%d",&num);                 //element to be
replaced
 arr[i][j][k]=num;                 //element assigned to
array position
 printf("\narray after updating: \n");
```

```
for(i=1;i<=2;i++)
 {
  for(j=1;j<=3;j++)
  {
   for(k=1;k<=3;k++)
   {
    printf("%d ",arr[i][j][k]);
    if(k==3)
    {
     printf("\n");
    }
   }
  }
  printf("\n");
 }
 return 0;
}
```

Output:

```
enter the values in the array:

the value at arr[1][1][1]: 11
the value at arr[1][1][2]: 22
the value at arr[1][1][3]: 33
the value at arr[1][2][1]: 44
the value at arr[1][2][2]: 55
the value at arr[1][2][3]: 66
the value at arr[1][3][1]: 77
the value at arr[1][3][2]: 88
```

the value at arr[1][3][3]: 99
the value at arr[2][1][1]: 10
the value at arr[2][1][2]: 20
the value at arr[2][1][3]: 30
the value at arr[2][2][1]: 40
the value at arr[2][2][2]: 50
the value at arr[2][2][3]: 60
the value at arr[2][3][1]: 70
the value at arr[2][3][2]: 80
the value at arr[2][3][3]: 90

printing the values in array:
11 22 33
44 55 66
77 88 99

10 20 30
40 50 60
70 80 90
enter the block row and column number:2 1 1
enter the new number you want to update with: 15

array after updating:
11 22 33
44 55 66
77 88 99

15 20 30
40 50 60
70 80 90

In the above program;

- We have declared three variables i,j,k for three for loops and num variable which will hold the value/element to be updated.
- we have declared an array of type integer int arr[2][3][3];(blocks:2 rows:3 columns:3)
- First section of nested for loops ask the user to insert the values.
- second section of nested for loops will print the inserted values in the matrix form.
- Position/value will be updated.
- Third section of nested for loop will print the updated 3D array.

- **Make a program that will Convert 3D array into 2D array.**

```
#include <iostream>
using namespace std;
int main()
{
 int i,j,k;              //variables for nested for loop
 int arr[2][3][3];       //declaration of 3D array
 printf("enter the elements: \n");
 for(i=0;i<2;i++)
 {
  for(j=0;j<3;j++)
  {
   for(k=0;k<3;k++)
   {
```

```
 printf("element at [%d][%d][%d]: ",i,j,k);
   scanf("%d",&arr[i][j][k]);
   //arr[i][j][k]=++ctr;
  }
 }
}
printf("\nprinting 3D array\n");
for(i=0;i<2;i++)                    //printing 3d array
{
 for(j=0;j<3;j++)
 {
  for(k=0;k<3;k++)
  {
   printf("%d ",arr[i][j][k]);
  }
  printf("\n");
 }
 printf("\n");
}
int a[3][3];                  //2D array declaration
int b[3][3];
printf("\ncopying values in new 2D array: \n");
for(i=0;i<2;i++)
{
 for(j=0;j<3;j++)
 {
  for(k=0;k<3;k++)
  {
```

```
if(i==0)
   {
    a[j][k]=arr[i][j][k];    //copying values in new 2d
array
   }
   else
   {
    b[j][k]=arr[i][j][k];
   }
  }
 }
}
 printf("\nprinting elements in first 2D array: \n");
 for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)                //printing first 2d array
  {
   printf("%d ",a[i][j]);
  }
  printf("\n");
 }
 printf("\n");
 printf("\nprinting elements in second 2D array: ");
 for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)                //printing 2nd 2d array
  {
   printf("%d ",b[i][j]);
  }
```

```
printf("\n");
 }
 return 0;
}
```

- **Make a program that will convert 2D array into 3D array**

```
#include <iostream>
using namespace std;
int main()
{
 int i,j,k;
 int a[3][3];                  //2D array declaration
 int b[3][3];
 printf("enter the elements in array a: \n");
//entering elements in array 'a'
 for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)
  {
   printf("element at [%d][%d]: ",i,j);
   scanf("%d",&a[i][j]);
  }
 }
 printf("\nprinting 2D array a\n");                //printing
elements of array 'a'
 for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)
  {
   printf("%d ",a[i][j]);
  }
```

```
 printf("\n");
 }
 printf("\nenter the elements in array b: \n");
//entering elements in array 'b'
 for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)
  {
   printf("element at [%d][%d]: ",i,j);
   scanf("%d",&b[i][j]);
  }
 }
 printf("\nprinting 2D array b\n");                //printing
elements of array 'b'
 for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)
  {
   printf("%d ",b[i][j]);
  }
  printf("\n");
 }
 int arr[3][3][3];                              //3D array
declaration
 printf("\ncopying values in 3D array: \n");
 for(i=0;i<2;i++)
 {
  for(j=0;j<3;j++)
  {
   for(k=0;k<3;k++)
```

```
 {
    if(i==0)
    {
     arr[i][j][k]=a[j][k];              //copying elements of
2D array into 3D array
    }
    else
    {
     arr[i][j][k]=b[j][k];
    }
   }
  }
 }
 printf("\nprinting elements in 3D array: \n");
//printing elements of 3D array
 for(i=0;i<2;i++)
 {
  for(j=0;j<3;j++)
  {
   for(k=0;k<3;k++)
   {
   printf("%d ",arr[i][j][k]);
   }
   printf("\n");
  }
 printf("\n");
 }
 return 0;
}
```

- **Dynamically allocating memory using malloc in 3D array.**

As we know that static array variables are fixed in size and can't be changed(enlarged or shrinked).To remove this drawback we use dynamic memory allocation.dynamic array is nothing but it is allocated during runtime with malloc or calloc.
*syntax: int *array=int(int *)malloc(sizeof(int)element-count);
Example :

```
#include <stdio.h>
#include <malloc.h>             //malloc library
int main(int argc, char* argv[]) //command line arguments
{
  int ***arr;                   //triple pointer
  int block,row,column;         //variables for block, rows and columns
  int i,j,k;                    //nested for loop
  printf("enter the blocks, rows and columns: ");
  scanf("%d %d %d",&block,&row,&column);
  arr=(int ***)malloc(sizeof(int ***)*block);
  for(i=0;i<block;i++) {
  arr[i]=(int **)malloc(sizeof(int*)*row);
```

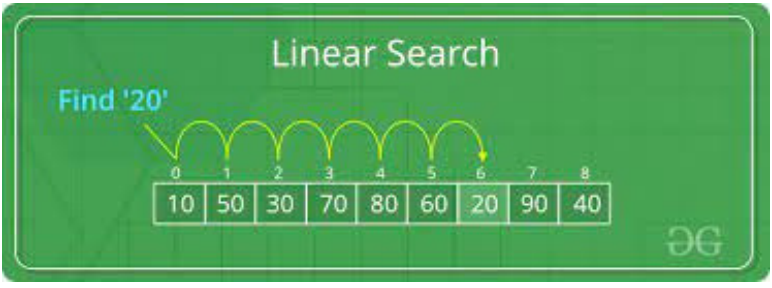
```
 for(j=0;j<row;j++)
{
    arr[i][j]=(int *)malloc(sizeof(int)*column);
  }
 }
 for(i=0;i<block;i++)
 {
  for(j=0;j<row;j++)
{
    for(k=0;k<column;k++)
 {
      printf("element at [%d][%d][%d] : ",i,j,k);
      scanf("%d",&arr[i][j][k]);
    }
  }
 }
 printf("Printing 3D Array:\n");
 for(i=0;i<block;i++)
 {
  for(j=0;j<row;j++)
{
    for(k=0;k<column;k++)
 {
      printf("%.2d ",arr[i][j][k]);
    }
    printf("\n");
  }
  printf("\n");
 }
 return 0;
}
```

- **Searching :**

1. Linear Search
2. Binary Search
3. Modified Binary Search
4. Binary Search on 2D Arrays

1. Linear Search :

Linear search is a simple searching algorithm that iterates through an array or a list to find a specific element. It sequentially checks each element until the target element is found or until the end of the list is reached.



Time Complexity:
Best case: O(1)
Worst case: O(n)

Where n is the size of the array.

Space Complexity:
O(1)

- **What is the output of the following program fragment:**

```
#include <stdio.h>

int linearSearch(int arr[], int size, int target) {
  for (int i = 0; i < size; i++) {
    if (arr[i] == target) {
      return i;
    }
  }
  return -1;
}

int main() {
  int numbers[] = {12, 45, 67, 23, 9, 8, 34, 78};
  int targetElement = 23;
  int size = sizeof(numbers) / sizeof(numbers[0]);

  int index = linearSearch(numbers, size,
targetElement);

  if (index != -1) {
    printf("Element %d found at index: %d\n",
targetElement, index);
  } else {
    printf("Element %d not found in the array.\n",
targetElement);
  }

  return 0;
}
```

Output:
Element 23 found at index: 3

- **What is the output of the following program fragment:**

```
#include <stdio.h>

#define MAX_VALUE 2147483647

int linearSearch(int arr[], int size, int target) {
 for (int i = 0; i < size; i++) {
  if (arr[i] == target) {
   return arr[i];
  }
 }
 return MAX_VALUE;
}

int main() {
 int numbers[] = {12, 45, 67, 23, 9, 8, 34, 78};
 int targetElement = 230;

 int element = linearSearch(numbers, sizeof(numbers) /
sizeof(numbers[0]), targetElement);

 if (element != MAX_VALUE) {
  printf("Element %d found\n", element);
 } else {
  printf("Element %d not found in the array.\n",
targetElement);
 }
```

```
 return 0;
}
```

Output:

Element 230 not found in the array.

- **What is the output of the following program fragment:**

```
#include <stdio.h>

int linearSearch(int arr[], int size, int target) {
  for (int index = 0; index < size; index++) {
    if (target == arr[index]) {
      return index;
    }
  }
  return -1;
}

int main() {
  int n;
  printf("How many numbers do you want to enter:
");
  scanf("%d", &n);

  int array[n];

  for (int i = 0; i < n; i++) {
    printf("Enter array[%d] = ", i);
    scanf("%d", &array[i]);
  }

  printf("Enter checking item: ");
  int targetNumber;
  scanf("%d", &targetNumber);
```

```
 int x = linearSearch(array, n, targetNumber);

 if (x != -1) {
 printf("The number %d found at %d\n",
targetNumber, x);
 } else {
 printf("Number doesn't found!\n");
 }

 return 0;
}
```

Output:

```
How many numbers do you want to enter: 5
Enter array[0] = 57
Enter array[1] = 9
Enter array[2] = 3
Enter array[3] = 88
Enter array[4] = 0
Enter checking item: 88
The number 88 found at 3
```

- **What is the output of the following program fragment:**

```
#include <stdio.h>

int linearSearch(int arr[], int length, int target) {
  for (int index = 0; index < length; index++) {
    if (target == arr[index]) {
      return index;
    }
  }
  return -1;
}

int main() {
  int length;
  printf("How many numbers do you want to enter: ");
  scanf("%d", &length);

  int array[length];

  for (int i = 0; i < length; i++) {
    printf("Enter array[%d] = ", i);
    scanf("%d", &array[i]);
  }

  printf("Enter checking item: ");
  int targetNumber;
  scanf("%d", &targetNumber);

  int x = linearSearch(array, length, targetNumber);
```

```
 if (x != -1) {
 printf("The number %d found at %d\n",
targetNumber, x);
 } else {
 printf("Number doesn't found!\n");
 }

 return 0;
}
```

Output:

```
How many number do you want to enter : 5
Enter array[0] = 2
Enter array[1] = 5
Enter array[2] = 8
Enter array[3] = 90
Enter array[4] = 3
Enter checking item : 90
The number 90 found at 3
```

- **What is the output of the following program fragment:**

```
#include <stdio.h>
#include <stdbool.h>

bool linearSearch(char str[], char target) {
 for (int i = 0; str[i] != '\0'; i++) {
  if (str[i] == target) {
   return true;
  }
 }
 return false;
}

int main() {
 char name[] = "Hridi";
 char target = 'r';

 bool bol = linearSearch(name, target);

 if (bol) {
  printf("%c match\n", target);
 } else {
  printf("%c not match\n", target);
 }

 return 0;
}
```

Output:

r match

- **What is the output of the following program fragment:**

```
#include <stdio.h>
#include <stdbool.h>

bool linearSearch(char str[], char target) {
 while (*str != '\0') {
  if (*str == target) {
   return true;
  }
  str++;
 }
 return false;
}

int main() {
 char name[] = "Hridi";
 char target = 'k';

 bool bol = linearSearch(name, target);

 if (bol) {
  printf("%c match\n", target);
 } else {
  printf("%c not match\n", target);
 }

 return 0;
}
```

Output:

k not match

- **What is the output of the following program fragment:**

```
#include <stdio.h>
#include <stdbool.h>
#include <string.h>

bool linearSearch(char str[], char target) {
 for (int i=0; str[i] != '\0'; i++) {
  if (str[i] == target) {
   return true;
  }
 }
 return false;
}

int main() {
 char name[] = "Hridi";
 char target = 'H';

 printf("%s\n", name);

 bool bol = linearSearch(name, target);

 if (bol) {
  printf("%c match\n", target);
 } else {
  printf("%c not match\n", target);
 }

 return 0;
}
```

Output:
[H, r, i, d, i]
k not match

- **What is the output of the following program fragment:**

```
#include <stdio.h>

int minimumNumber(int arr[], int size) {
  int temp = arr[0];
  for (int i = 1; i < size; i++) {
    if (temp > arr[i]) {
      temp = arr[i];
    }
  }
  return temp;
}

int main() {
  int numbers[] = {12, 45, 67, 23, 9, 8, 34, 78};
  int size = sizeof(numbers) / sizeof(numbers[0]);

  printf("Minimum number = %d\n",
minimumNumber(numbers, size));

  return 0;
}
```

Output:
Minimum number = 8

- **What is the output of the following program fragment:**

```
#include <stdio.h>

int maximumNumber(int arr[], int size) {
  int temp = arr[0];
  for (int i = 1; i < size; i++) {
    if (temp < arr[i]) {
      temp = arr[i];
    }
  }
  return temp;
}

int main() {
  int numbers[] = {12, 45, 67, 23, 9, 8, 34, 78};
  int size = sizeof(numbers) / sizeof(numbers[0]);

  printf("Maximum number = %d\n",
maximumNumber(numbers, size));

  return 0;
}
```

Output:
Maximum number = 78

- **What is the output of the following program fragment:**

```
#include <stdio.h>

#define ROWS 3
#define COLS 3

void linearSearch2DArray(int arr[ROWS][COLS], int
target, int result[]) {
  for (int row = 0; row < ROWS; row++) {
    for (int col = 0; col < COLS; col++) {
      if (arr[row][col] == target) {
        result[0] = row;
        result[1] = col;
        return;
      }
    }
  }
  result[0] = -1;
  result[1] = -1;
}

int main() {
  int matrix[ROWS][COLS] = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
  };
  int targetElement = 2;
  int result[2];
```

```
 linearSearch2DArray(matrix, targetElement,
result);
 if (result[0] != -1 && result[1] != -1) {
 printf("Element %d found at row: %d, col: %d\n",
targetElement, result[0], result[1]);
 } else {
 printf("Element %d not found in the 2D array.\n",
targetElement);
 }
 return 0;
}
```

Output:
Element 5 found at row: 1, col: 1

- **What is the output of the following program fragment:**

```
#include <stdio.h>

#define ROWS 4
#define MAX_COLS 4

void linearSearch2DArray(int arr[ROWS][MAX_COLS],
int target, int result[]) {
    for (int row = 0; row < ROWS; row++) {
        for (int col = 0; col < MAX_COLS; col++) {
            if (arr[row][col] == target) {
                result[0] = row;
                result[1] = col;
                return;
            }
        }
    }
```

```
 result[0] = -1;
 result[1] = -1;
}

int main() {
 int matrix[ROWS][MAX_COLS] = {
 {23, 4, 1},
 {18, 12, 3, 9},
 {78, 99, 34, 56},
 {18, 12}
 };
 int targetElement = 34;
 int result[2];

 linearSearch2DArray(matrix, targetElement,
result);

 printf("[%d, %d]\n", result[0], result[1]);

 return 0;
}
```

Output:

[2, 2]

- **What is the output of the following program fragment:**

```
#include <stdio.h>
#define ROWS 4
#define MAX_COLS 4

int MaxValue(int arr[ROWS][MAX_COLS]) {
   int max = -2147483648; // INT_MIN

   for (int row = 0; row < ROWS; row++) {
      for (int col = 0; col < MAX_COLS; col++) {
         if (arr[row][col] > max) {
            max = arr[row][col];
         }
      }
   }
   return max;
}
int main() {
   int matrix[ROWS][MAX_COLS] = {
      {23, 4, 1},
      {18, 12, 3, 9},
      {78, 99, 34, 56},
      {18, 12}
   };

   int result = MaxValue(matrix);
   printf("%d\n", result);
   return 0;
}
```

Output:

99

- **What is the output of the following program fragment:**

```
#include <stdio.h>
#include <limits.h>

#define ROWS 4
#define MAX_COLS 4

int MinValue(int arr[ROWS][MAX_COLS]) {
  int min = INT_MAX;

  for (int row = 0; row < ROWS; row++) {
    for (int col = 0; col < MAX_COLS; col++) {
      if (arr[row][col] < min) {
        min = arr[row][col];
      }
    }
  }

  return min;
}

int main() {
  int matrix[ROWS][MAX_COLS] = {
    {23, 4, 1, 32},
    {18, 12, 3, 9},
    {78, 99, 34, 56},
    {18, 12, 55, 6}
  };
```

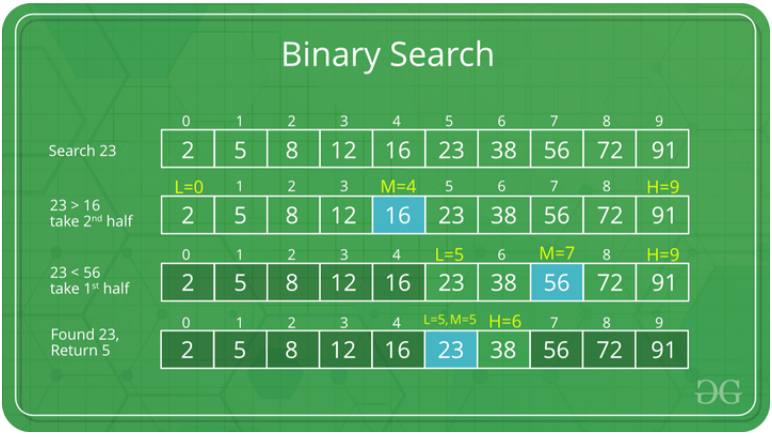
```
  int result = MinValue(matrix);

  printf("%d\n", result);

  return 0;
}
```

Output:

```
1
```

2. Binary Search :

Binary search is a search algorithm that finds the position of a target value within a sorted array. It works by repeatedly dividing the search space in half until the target value is found or it is determined that the target value does not exist in the array.



Time Complexity:
Best case: O(1)
Worst case: O(log n)

Where n is the size of the array.

Space Complexity:
O(1)

- **What is the output of the following program fragment:**

```
#include <iostream>
#include <vector>

using namespace std;

int binarySearch(int arr[], int target, int size) {
  int left = 0;
  int right = size - 1;

  while (left <= right) {
    int mid = left + (right - left) / 2;

    if (arr[mid] == target) {
      return mid;
    }

    if (arr[mid] > target) {
      right = mid - 1;
    } else {
      left = mid + 1;
    }
  }

  return -1;
}
```

```
int main() {
 int sortedArray[] = {1, 3, 5, 7, 9, 11, 13, 15, 17};
 int target = 11;
 int index = binarySearch(sortedArray, target,
sizeof(sortedArray)/sizeof(sortedArray[0]));

 if (index != -1) {
 cout << "Target found at index: " << index << endl;
 } else {
 cout << "Target not found in the array." << endl;
 }

 return 0;
}
```

Output:

Target found at index: 5

- **Ascending order:**

- **What is the output of the following program fragment:**

```
#include <iostream>

using namespace std;

int binarySearch(const int arr[], int size, int target) {
  int left = 0;
  int right = size - 1;

  while (left <= right) {
    int mid = left + (right - left) / 2;

    if (target < arr[mid]) {
      right = mid - 1;
    } else if (target > arr[mid]) {
      left = mid + 1;
    } else {
      return mid;
    }
  }

  return -1;
```

```
int main() {
 int sortedArray[] = {1, 3, 5, 7, 9, 11, 13, 15, 17};
 int size = sizeof(sortedArray) /
sizeof(sortedArray[0]);
 int target = 11;
 int index = binarySearch(sortedArray, size, target);

 if (index != -1) {
 cout << "Target found at index: " << index << endl;
 } else {
 cout << "Target not found in the array." << endl;
 }

 return 0;
}
```

Output:

Target found at index: 5

- **Descending order:**

- **What is the output of the following program fragment:**

```
#include <iostream>

using namespace std;

int binarySearch(int arr[], int size, int target) {
  int left = 0;
  int right = size - 1;

  while (left <= right) {
    int mid = left + (right - left) / 2;

    if (target < arr[mid]) {
      left = mid + 1;
    } else if (target > arr[mid]) {
      right = mid - 1;
    } else {
      return mid;
    }
  }

  return -1;
}
```

```
int main() {
 int sortedArray[] = {17, 15, 13, 11, 9, 7, 5, 3, 1};
 int size = sizeof(sortedArray) / sizeof(sortedArray[0]);
 int target = 11;
 int index = binarySearch(sortedArray, size, target);

 if (index != -1) {
 cout << "Target found at index: " << index << endl;
 } else {
 cout << "Target not found in the array." << endl;
 }

 return 0;
}
```

Output:

Target found at index: 3

- **When we don't know ascending or descending order:**

- **What is the output of the following program fragment:**

```
#include <iostream>

using namespace std;

int binarySearch(const int arr[], int target, int size) {
  int left = 0;
  int right = size - 1;

  bool isAsc = arr[left] < arr[right];

  while (left <= right) {
    int mid = left + (right - left) / 2;

    if (arr[mid] == target) {
      return mid;
    }

    if (isAsc) {
      if (arr[mid] > target) {
        right = mid - 1;
      } else {
        left = mid + 1;
      }
```

```
 } else {
 if (arr[mid] < target) {
 right = mid - 1;
 } else {
 left = mid + 1;
 }
 }
 }

 return -1;
}

int main() {
 int sortedArray[] = {1, 3, 5, 7, 9, 11, 13, 15, 17};
 int size = sizeof(sortedArray) /
sizeof(sortedArray[0]);
 int target = 11;
 int index = binarySearch(sortedArray, target, size);

 if (index != -1) {
 cout << "Target found at index: " << index << endl;
 } else {
 cout << "Target not found in the array." << endl;
 }

 return 0;
}
```

Output:

Target found at index: 5

- **Binary search(First occurrence) from sorted array.**

```
#include <iostream>

using namespace std;

int binarySearch(int arr[], int target, int size) {
  int left = 0;
  int right = size - 1;

  while (left <= right) {
    int mid = left + (right - left) / 2;

    if (arr[mid] == target) {
      if (mid == 0 || arr[mid - 1] < target) {
        return mid;
      } else {
        right = mid - 1;
      }
    } else if (arr[mid] < target) {
      left = mid + 1;
    } else {
      right = mid - 1;
    }
  }

  return -1;
}
```

```
int main() {
 int arr[] = {1, 2, 2, 2, 3, 4, 4, 5, 6};
 int size = sizeof(arr) / sizeof(arr[0]);
 int target = 4;

 int result = binarySearch(arr, target, size);

 if (result != -1) {
 cout << "First occurrence of " << target << " is at
index " << result << endl;
 } else {
 cout << target << " not found in the array" << endl;
 }

 return 0;
}
```

Output:

First occurrence of 4 is at index 5

- **Binary search(Last occurrence) from sorted array.**

```
#include <iostream>

using namespace std;

int binarySearch(int arr[], int target, int size) {
    int left = 0;
    int right = size - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == target) {
            if (mid == size - 1 || arr[mid + 1] > target) {
                return mid;
            } else {
                left = mid + 1;
            }
        } else if (arr[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    return -1;
}
```

```
int main() {
 int arr[] = {1, 2, 2, 2, 3, 4, 4, 5, 6};
 int size = sizeof(arr) / sizeof(arr[0]);
 int target = 4;

 int result = binarySearch(arr, target, size);

 if (result != -1) {
 cout << "First occurrence of " << target << " is at
index " << result << endl;
 } else {
 cout << target << " not found in the array" << endl;
 }

 return 0;
}
```

Output:

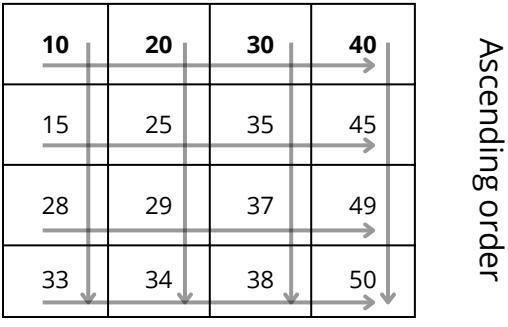
Last occurrence of 4 is at index 6

- **Binary Search in 2D Arrays:**

Type 1 : Unsorted 2D

Rows and columns are in ascending order.

Ascending order



Target : 37

- **What is the output of the following program fragment:**

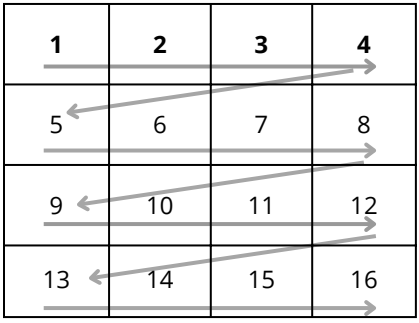
```
#include <iostream>

using namespace std;

int* binarySearchIn2DArray(int matrix[][4], int
rows, int cols, int target) {
  int r = 0;
  int c = cols - 1;

  while (r < rows && c >= 0) {
    if (matrix[r][c] == target) {
      int* result = new int[2];
      result[0] = r;
      result[1] = c;
      return result;
    }
    if (matrix[r][c] < target) {
      r++;
    } else {
      c--;
    }
  }

  int* result = new int[2];
  result[0] = -1;
  result[1] = -1;
  return result;
}
```

```
int main() {
 int arr[][4] = {
 {10, 20, 30, 40},
 {15, 25, 35, 45},
 {28, 29, 37, 49},
 {33, 34, 38, 50}
 };

 int* result = binarySearchIn2DArray(arr, 4, 4, 37);

 cout << "[" << result[0] << ", " << result[1] << "]" <<
endl;

 delete[] result; // Remember to free the memory
allocated with 'new'

 return 0;
}
```

Output:

```
[2, 2]
```

Type 2 : Sorted 2D



Target : 3

- **What is the output of the following program fragment:**

```
#include <iostream>

using namespace std;

// Function to perform binary search in a specific row within a given column range
int* binarySearch(int** matrix, int row, int cStart, int cEnd, int target) {
  while (cStart <= cEnd) {
    int mid = cStart + (cEnd - cStart) / 2;
    if (matrix[row][mid] == target) {
      // Target found in the current row and column
      int* result = new int[2]{row, mid};
      return result;
    }
```

```
    if (matrix[row][mid] < target) {
      cStart = mid + 1;
    } else {
      cEnd = mid - 1;
    }
  }
  int* result = new int[2]{-1, -1}; // Target not found in the
current row and column range
  return result;
}

// Function to perform binary search in a 2D array
int* binarySearchIn2DArray(int** matrix, int rows, int
cols, int target) {
  if (rows == 1) {
    // Single row, perform binary search in that row
    return binarySearch(matrix, 0, 0, cols - 1, target);
  }

  int rStart = 0;
  int rEnd = rows - 1;
  int cMid = cols / 2;

  // Run the loop until 2 rows are remaining
  while (rStart < (rEnd - 1)) {
    int mid = rStart + (rEnd - rStart) / 2;

    if (matrix[mid][cMid] == target) {
      // Target found in the middle column
      int* result = new int[2]{mid, cMid};
      return result;
    }
```

```
 if (matrix[mid][cMid] < target) {
 rStart = mid + 1;
 } else {
 rEnd = mid - 1;
 }
 }

  // Now we have two rows
  // Check whether the target is in the column of these
two rows

  if (matrix[rStart][cMid] == target) {
    int* result = new int[2]{rStart, cMid};
    return result;
  }
  if (matrix[rStart + 1][cMid] == target) {
    int* result = new int[2]{rStart + 1, cMid};
    return result;
  }

  // Perform binary search in the 1st half
  if (target <= matrix[rStart][cMid - 1]) {
    return binarySearch(matrix, rStart, 0, cMid - 1, target);
  }
  // Perform binary search in the 2nd half
  if (target >= matrix[rStart][cMid + 1]) {
    return binarySearch(matrix, rStart, cMid + 1, cols - 1,
target);
  }
  // Perform binary search in the 3rd half
  if (target <= matrix[rStart + 1][cMid - 1]) {
```

```
 return binarySearch(matrix, rStart + 1, 0, cMid - 1,
target);
 }
 // Perform binary search in the 4th half
 if (target >= matrix[rStart + 1][cMid + 1]) {
 return binarySearch(matrix, rStart + 1, cMid + 1, cols -
1, target);
 }

 int* result = new int[2]{-1, -1}; // Target not found
 return result;
}

int main() {
 int rows = 4;
 int cols = 4;

 int** arr = new int*[rows];
 for (int i = 0; i < rows; ++i) {
 arr[i] = new int[cols];
 }

 // Initialize the 2D array
 int counter = 1;
 for (int i = 0; i < rows; ++i) {
 for (int j = 0; j < cols; ++j) {
 arr[i][j] = counter++;
 }
 }
```

```
    int* result = binarySearchIn2DArray(arr, rows, cols, 3);

    // Print the result
    cout << "[" << result[0] << ", " << result[1] << "]" << endl;

    // Clean up dynamically allocated memory
    for (int i = 0; i < rows; ++i) {
        delete[] arr[i];
    }
    delete[] arr;
    delete[] result;

    return 0;
}
```

Output:

[0, 2]

Type 3 : Assuming the 2D Array as 1D

2	**4**	**6**	**8**
10	12	14	16
18	20	22	24
26	28	30	32

Target : 20

- **What is the output of the following program fragment:**

```
#include <iostream>
#include <vector>

using namespace std;

vector<int> binarySearchIn2DArray(vector<vector<int>>&
matrix, int target) {
  int rows = matrix.size();
  int cols = matrix[0].size();
  int left = 0;
  int right = rows * cols - 1;
```

```
while (left <= right) {
    int mid = left + (right - left) / 2;
    int midValue = matrix[mid / cols][mid % cols];

    if (midValue == target) {
      return {mid / cols, mid % cols};
    } else if (midValue < target) {
      left = mid + 1;
    } else {
      right = mid - 1;
    }
  }

  return {-1, -1};
}

int main() {
  vector<vector<int>> sortedMatrix = {
    {2, 4, 6, 8},
    {10, 12, 14, 16},
    {18, 20, 22, 24},
    {26, 28, 30, 32}
  };

int target = 20;

  vector<int> result =
binarySearchIn2DArray(sortedMatrix, target);
  int row = result[0];
  int col = result[1];
```

```
  if (row != -1 && col != -1) {
    cout << "Target found at row " << row << " and
column " << col << endl;
  } else {
    cout << "Target not found in the 2D array." << endl;
  }

  return 0;
}
```

Output:

Target found at row 2 and column 1

- **Sorting :**

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Cycle Sort

- **Inbuilt Sort in C++ :**

- **What will be the output of the following code?**

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
  int n;
  cin>>n;
  int a[n];
  for(int i=0; i<n; i++){
    cin>> a[i];
  }

  sort(a, a+n);

  for(int i=0; i<n; i++){
    cout<< a[i] <<" ";
  }
}
```

Output:

```
5
2 4 1 5 3
1 2 3 4 5
```

- **What will be the output of the following code?**

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
  int n;
  cin>>n;
  int a[n];
  for(int i=0; i<n; i++){
    cin>> a[i];
  }

  sort(a+2, a+n);

  for(int i=0; i<n; i++){
    cout<< a[i] <<" ";
  }
}
```

Output:

```
5
2 4 1 5 3
2 4 1 3 5
```

- **What will be the output of the following code?**

```
#include <iostream>
#include <algorithm> // for std::sort

using namespace std;

int main() {
    // Create an array of integers
    int numbers[] = {5, 2, 8, 1, 6};

    // Use sort to sort the array in ascending order
    sort(begin(numbers), end(numbers));

    cout << "Sorted Array (Ascending Order): ";
    for (const auto& num : numbers) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

Output:
Sorted Array (Ascending Order): 1 2 5 6 8

- **What will be the output of the following code?**

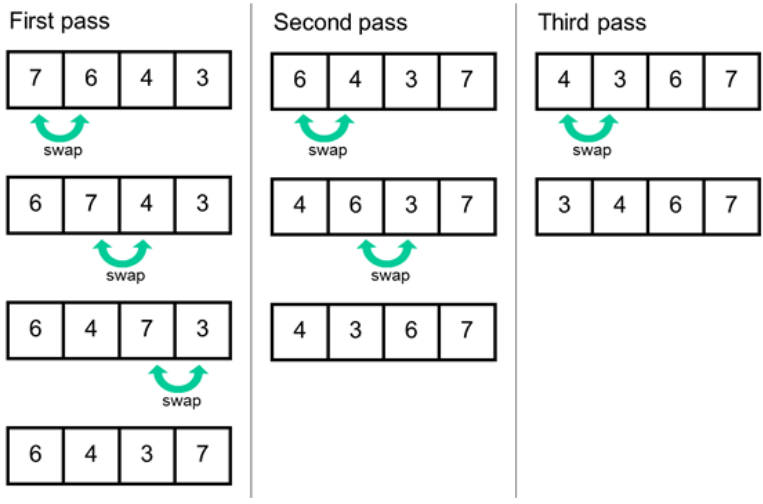
```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
  // Create a vector of integers
  vector<int> numbers = {5, 2, 8, 1, 6};

  // Use sort to sort the vector in ascending order
  sort(numbers.begin(), numbers.end());

  cout << "Sorted Vector (Ascending Order): ";
  for (const auto& num : numbers) {
    cout << num << " ";
  }
  cout << endl;

  // Use sort to sort the vector in descending order
  sort(numbers.rbegin(), numbers.rend());

  cout << "Sorted Vector (Descending Order): ";
  for (const auto& num : numbers) {
    cout << num << " ";
  }
  cout << endl;

  return 0;
}
```

Output:

Sorted Vector (Ascending Order): 1 2 5 6 8
Sorted Vector (Descending Order): 8 6 5 2 1

1. Bubble Sort :

Bubble sort is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares adjacent elements, and swaps them if they are in the wrong order. The process is repeated until the entire list is sorted.



Time Complexity:
Best case: O(n)
Worst case: O(n^2)

Where n is the size of the array.

Space Complexity:
O(1)

- **What is the output of the following program fragment:**

```
#include <iostream>
#include <algorithm>

using namespace std;

void bubbleSort(int arr[], int size) {
  bool swapped;

  for (int i = 0; i < size - 1; i++) {
    swapped = false;

    for (int j = 0; j < size - 1 - i; j++) {
      if (arr[j] > arr[j + 1]) {
        // Swap arr[j] and arr[j+1]
        int temp = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = temp;
        swapped = true;
      }
    }
    // If no two elements were swapped in the inner
loop, the array is already
    // sorted
    if (!swapped) {
      break;
    }
  }
}
```

```
int main() {
 int arr[] = {64, 34, 25, 12, 22, 11, 90};
 int size = sizeof(arr) / sizeof(arr[0]);

 cout << "Original array: ";
 for (int i = 0; i < size; i++) {
 cout << arr[i] << " ";
 }
 cout << endl;

 bubbleSort(arr, size);

 cout << "Sorted array: ";
 for (int i = 0; i < size; i++) {
 cout << arr[i] << " ";
 }
 cout << endl;

 return 0;
}
```

Output:

Original array: 64 34 25 12 22 11 90
Sorted array: 11 12 22 25 34 64 90

- **What is the output of the following program fragment:**

```
#include <iostream>
#include <algorithm>

using namespace std;

void bubbleSort(int arr[], int size) {
  bool swapped;

  for (int i = 0; i < size - 1; i++) {
    swapped = false;

    for (int j = 0; j < size - i - 1; j++) {
      if (arr[j] < arr[j + 1]) {
        // Swap arr[j] and arr[j+1]
        int temp = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = temp;
        swapped = true;
      }
    }
    // If no two elements were swapped in the inner loop,
the array is already sorted
    if (!swapped) {
      break;
    }
  }
}
```

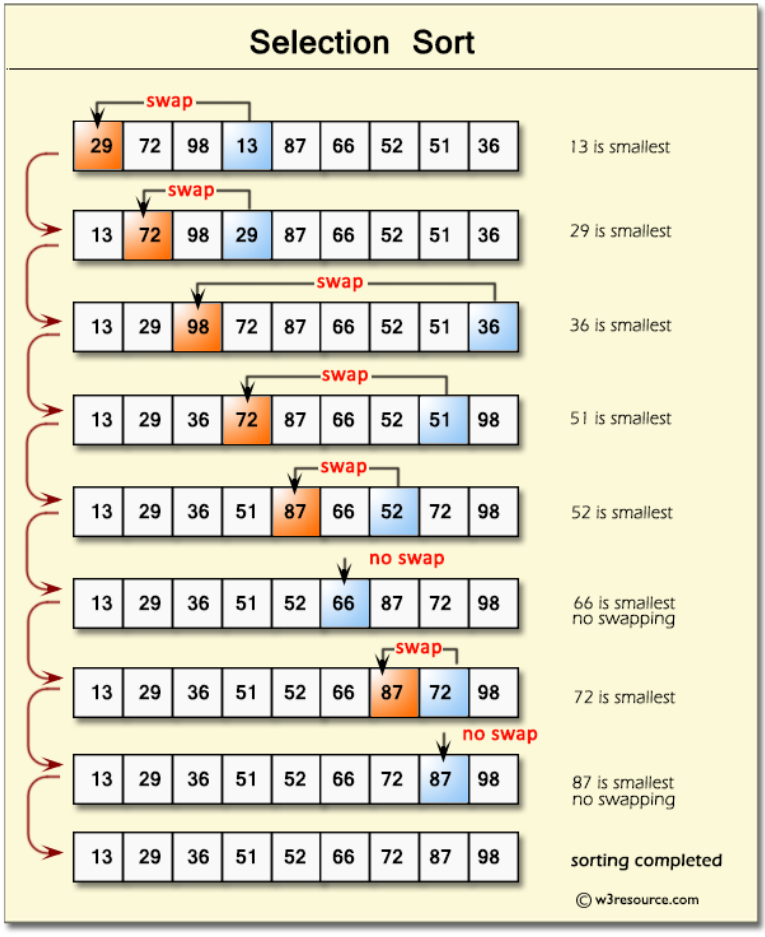
```
int main() {
 int arr[] = {64, 34, 25, 12, 22, 11, 90};
 int size = sizeof(arr) / sizeof(arr[0]);

 cout << "Original array: " << "[";
 for (int i = 0; i < size; i++) {
 cout << arr[i];
 if (i < size - 1) {
 cout << ", ";
 }
 }
 cout << "]" << endl;

 bubbleSort(arr, size);

 cout << "Sorted array: " << "[";
 for (int i = 0; i < size; i++) {
 cout << arr[i];
 if (i < size - 1) {
 cout << ", ";
 }
 }
 cout << "]" << endl;

 return 0;
}
```

Output:

Original array: [64, 34, 25, 12, 22, 11, 90]
Sorted array: [90, 64, 34, 25, 22, 12, 11]

2. Selection Sort :

Selection Sort is a simple sorting algorithm that works by repeatedly finding the minimum element from the unsorted part of the array and swapping it with the first unsorted element. It effectively divides the array into two parts: the sorted part on the left and the unsorted part on the right.



Time Complexity:
Best case: O(n^2)
Worst case: O(n^2)

Where n is the size of the array.

Space Complexity:
O(1)

It is not a stable algorithm.

- **What is the output of the following program fragment:**

```
#include <iostream>
#include <algorithm>

using namespace std;

void selectionSort(int arr[], int size) {
  for (int i = 0; i < size - 1; i++) {
    int minIndex = i;

    for (int j = i + 1; j < size; j++) {
      if (arr[j] < arr[minIndex]) {
        minIndex = j;
      }
    }

    // Swap the minimum element with the first
unsorted element
    int temp = arr[i];
    arr[i] = arr[minIndex];
    arr[minIndex] = temp;
  }
}

int main() {
  int arr[] = {64, 34, 25, 12, 22, 11, 90};
  int size = sizeof(arr) / sizeof(arr[0]);
```

```
 cout << "Original array: " << "[";
 for (int i = 0; i < size; i++) {
 cout << arr[i];
 if (i < size - 1) {
 cout << ", ";
 }
 }
 cout << "]" << endl;

 selectionSort(arr, size);

 cout << "Sorted array: " << "[";
 for (int i = 0; i < size; i++) {
 cout << arr[i];
 if (i < size - 1) {
 cout << ", ";
 }
 }
 cout << "]" << endl;

 return 0;
}
```

Output:

Original array: [64, 34, 25, 12, 22, 11, 90]
Sorted array: [11, 12, 22, 25, 34, 64, 90]

- **What is the output of the following program fragment:**

```
#include <iostream>
#include <algorithm>

using namespace std;

void selectionSortDescending(int arr[], int size) {
  for (int i = 0; i < size - 1; i++) {
    int maxIndex = i;

    for (int j = i + 1; j < size; j++) {
      if (arr[j] > arr[maxIndex]) {
        maxIndex = j;
      }
    }

    // Swap the maximum element with the first
unsorted element
    int temp = arr[i];
    arr[i] = arr[maxIndex];
    arr[maxIndex] = temp;
  }
}

int main() {
  int arr[] = {64, 34, 25, 12, 22, 11, 90};
  int size = sizeof(arr) / sizeof(arr[0]);
```

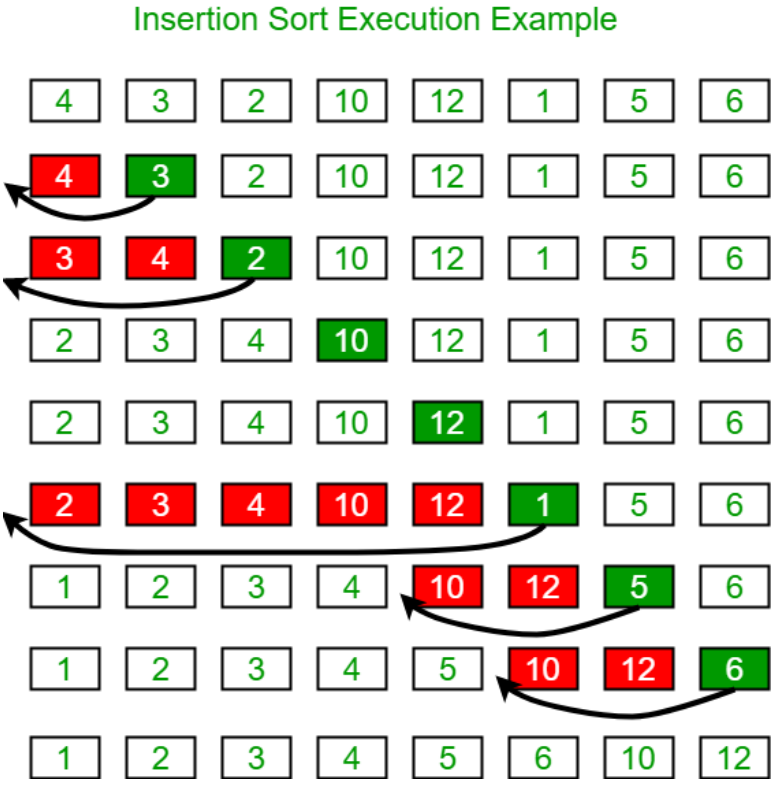
```
 cout << "Original array: " << "[";
 for (int i = 0; i < size; i++) {
 cout << arr[i];
 if (i < size - 1) {
 cout << ", ";
 }
 }
 cout << "]" << endl;

 selectionSortDescending(arr, size);

 cout << "Sorted array in descending order: " << "[";
 for (int i = 0; i < size; i++) {
 cout << arr[i];
 if (i < size - 1) {
 cout << ", ";
 }
 }
 cout << "]" << endl;

 return 0;
}
```

Output:

Original array: [64, 34, 25, 12, 22, 11, 90]
Sorted array in descending order: [90, 64, 34, 25, 22, 12, 11]

3. Insertion Sort :

The Insertion Sort algorithm works by iterating through the array and "inserting" each element into its correct position in the already sorted part of the array.



Time Complexity:
Best case: O(n)
Worst case: O(n^2)

Where n is the size of the array.

Space Complexity:
O(1)

- **What is the output of the following program fragment:**

```
#include <iostream>
#include <algorithm>

using namespace std;

void insertion(int arr[], int size) {
  for (int i = 0; i < size - 1; i++) {
    for (int j = i + 1; j > 0; j--) {
      if (arr[j - 1] > arr[j]) {
        // Swap arr[j] and arr[j-1]
        int temp = arr[j];
        arr[j] = arr[j - 1];
        arr[j - 1] = temp;
      } else {
        break;
      }
    }
  }
}

int main() {
  int arr[] = {0, -23, 56, 18};
  int size = sizeof(arr) / sizeof(arr[0]);

  insertion(arr, size);

  cout << "[";
```

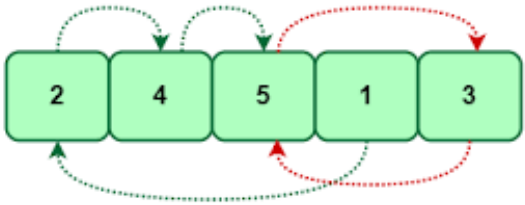
```
  for (int i = 0; i < size; i++) {
    cout << arr[i];
    if (i < size - 1) {
      cout << ", ";
    }
  }
  cout << "]" << endl;

  return 0;
}
```

Output:

```
[-23, 0, 18, 56]
```

4. Cycle Sort :

Cycle Sort is an in-place, unstable sorting algorithm that minimizes the number of writes performed during sorting. It is particularly useful for situations where writes to the data are more expensive than reads. The basic idea of Cycle Sort is to rotate the elements to their correct positions one by one, minimizing the number of swaps and writes.



Time Complexity:
Best case: O(n)
Worst case: O(n)

Where n is the size of the array.

Space Complexity:
O(1)

- **What is the output of the following program fragment:**

```
#include <iostream>
using namespace std;

void cycle(int arr[], int size) {
  int i = 0;
  while (i < size) {
    if (arr[i] != i + 1) {
      int temp = arr[arr[i] - 1];
      arr[arr[i] - 1] = arr[i];
      arr[i] = temp;
    } else {
      i++;
    }
  }
}

int main() {
  int arr[] = {5, 2, 4, 3, 1};
  int size = sizeof(arr) / sizeof(arr[0]);

  cycle(arr, size);

  cout << "[";
  for (int i = 0; i < size; i++) {
    cout << arr[i];
    if (i < size - 1) {
      cout << ", ";
    }
  }
```

```
  cout << "]" << endl;

  return 0;
}
```

Output:
[1, 2, 3, 4, 5]

- **What is the output of the following program fragment:**

```
#include <iostream>
using namespace std;

void cycle(int arr[], int size) {
  for (int i = 0; i < size;) {
    if (arr[i] != i + 1) {
      int temp = arr[arr[i] - 1];
      arr[arr[i] - 1] = arr[i];
      arr[i] = temp;
    } else {
      i++;
    }
  }
}

int main() {
  int arr[] = {5, 2, 4, 3, 1};
  int size = sizeof(arr) / sizeof(arr[0]);

  cycle(arr, size);
```

```
  cout << "[";
  for (int i = 0; i < size; i++) {
    cout << arr[i];
    if (i < size - 1) {
      cout << ", ";
    }
  }
  cout << "]" << endl;

  return 0;
}
```

Output:

```
[1, 2, 3, 4, 5]
```

- **What is the output of the following program fragment:**

```
#include <iostream>
using namespace std;

void cycle(int arr[], int size) {
  for (int i = 0; i < size;) {
    if ((arr[i] - 1) != i) {
      int temp = arr[arr[i] - 1];
      arr[arr[i] - 1] = arr[i];
      arr[i] = temp;
    } else {
      i++;
    }
  }
}

int main() {
  int arr[] = {5, 2, 4, 3, 1};
  int size = sizeof(arr) / sizeof(arr[0]);
  cycle(arr, size);
  cout << "[";
  for (int i = 0; i < size; i++) {
    cout << arr[i];
    if (i < size - 1) {
      cout << ", ";
    }
  }
  cout << "]" << endl;
  return 0;
}
```

Output:
[1, 2, 3, 4, 5]

- **What is the output of the following program fragment:**

```
#include <iostream>
using namespace std;

int cycle(int arr[], int size) {
  int i = 0;
  while (i < size) {
    if (arr[i] < size && arr[i] != arr[arr[i]]) {
      int temp = arr[arr[i]];
      arr[arr[i]] = arr[i];
      arr[i] = temp;
    } else {
      i++;
    }
  }

  for (int index = 0; index < size; index++) {
    if (arr[index] != index) {
      return index;
    }
  }

  return size;
}

int main() {
  int arr[] = {5, 2, 4, 0, 1};
  int size = sizeof(arr) / sizeof(arr[0]);
```

```
 int missingNum = cycle(arr, size);

 cout << "Missing number is " << missingNum << endl;

 return 0;
}
```

Output:
Missing number is 3

- **What is the output of the following program fragment:**

```
#include <iostream>

using namespace std;

void cycle(int arr[], int size) {
  for (int i = 0; i < size;) {
    if (arr[i] < size && arr[i] != i) {
      int temp = arr[arr[i]];
      arr[arr[i]] = arr[i];
      arr[i] = temp;
    } else {
      i++;
    }
  }
}

int main() {
  int arr[] = {5, 2, 0, 3, 1};
  int size = sizeof(arr) / sizeof(arr[0]);
```

```
 cycle(arr, size);

 cout << "[";
 for (int i = 0; i < size; i++) {
 cout << arr[i];
 if (i < size - 1) {
 cout << ", ";
 }
 }
 cout << "]" << endl;

 return 0;
}
```

Output:

[0, 1, 2, 3, 5]

4.strings

A String is a sequence of characters.

Example:

"This is a string."
"a"
" "

String representation :

There are 2 ways of string representation in C++.

1. The C style character string,
2. string class.

1. **C style character string :**

Strings are actually one-dimensional array of characters terminated by a null character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a null.

- **Make a program that will show your name using strings.**

```
#include <iostream>
#include<stdio.h>
using namespace std;
int main()
{
 char c[6];
 c[0]='H';
 c[1]='a';
 c[2]='m';
 c[3]='i';
 c[4]='m';
 c[5]='\0';
 printf("c = %s\n",c);
 return 0;
}
```

or,

```
#include <iostream>
#include<stdio.h>
using namespace std;
int main()
{
 char c[6]={'H','a','m','i','m','\0'};
 printf("c = %s\n",c);
 return 0;
}
```

Output:

c = Hamim

- **What is the output of the following C++ program fragment:**

```
#include<iostream>
using namespace std;
int main()
{
 char message1[6] = {'H','e','l','l','o'};
 char message2[] = {'H','e','l','l','o','\0'};
 char message3[]= "Hello";
 cout<<message1<<endl;
 cout<<message2<<endl;
 cout<<message3<<endl;
 return 0;
}
```

Output:
Hello
Hello
Hello

- **What is the output of the following C++ program fragment:**

```
#include<iostream>
using namespace std;
int main()
{
 char message1[6] = {'H','e','l','l','o'};
 char message2[] = {'H','e','l','l','o','\0'};
 char message3[]= "Hello";
 cout<<message1[0]<<endl;

  cout<<message2[1]<<endl;
  cout<<message3[2]<<endl;
  return 0;
}
```

Output:
H
e
l

- **What will be the output of given bellow.**

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main()
{
  char c[6] = {'H', 'a', 'm', 'i', 'm', '\0'};
  printf("c = %c\n", c[0]);
  return 0;
}
```

Output:

```
c = H
```

- **Make a program that will show your short name using strings.**

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main()
{
 char c[30];
 printf("Enter your name : ");
 scanf("%s",&c);
 printf("Your name is %s\n",c);
 return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
  char name[30];
  cout<<"Enter your name: ";
  cin>>name;
  cout<<"Your name is "<<name<<endl;
}
```

- **Make a program that will show your full name using strings.**

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main()
{
 char c[ ]="Hamim Talukder";
 printf("c = %s\n",c);
 return 0;
}
```

or,

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main()
{
 char c[15]="Hamim Talukder";
 printf("c = %s\n",c);
 return 0;
}
```

Here,
We also need a character variable for giving space between Hamim and Talukder.

- **Make a program that will show your full name using strings.**

```
#include<iostream>
using namespace std;
int main(){
  char name[15];
   cout<<"Enter your name : ";
   scanf(" %[^\n]",&name);
   cout<<"Your name is : "<<name<<endl;
   printf("Your full name is %s\n",name);
}
```

Output:

Enter your name : Hamim Talukder
Your name is : Hamim Talukder
Your full name is Hamim Talukder

Here,
In scanf() function %[^\n] can scan hole string including character space and set it into the character array's variable.

- **What will be the output of given bellow.**

```
#include <iostream>
using namespace std;

int main() {
   char name[15];
   cout << "Enter your name: ";
   cin.getline(name, sizeof(name));

   cout << "Your name is: " << name << endl;
   return 0;
}
```

Output:
Enter your name: Hamim talukder
Your name is: Hamim talukder

- **What will be the output of given bellow.**

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main()
{
 char c[30];
 printf("Enter your full name : ");
 gets(c);
 printf("Your full name is %s\n",c);
 return 0;
}
```

or,

```
#include <iostream>
#include <stdio.h>
#include<string.h>
using namespace std;
int main()
{
 char c[]="Hamim Talukder";
 printf("Enter your full name : ");
 gets(c);
 printf("Your full name is %s\n",c);
 return 0;
}
```

Output:

Enter your full name : Hamim Talukder
Your full name is Hamim Talukder

Here,
gets() function works as scanf() function do. Here scanf() function can not diagnosis the separate word after space but gets() function can do.

- **Make a program that will show your full name and the first letter of your name using strings.**

```
#include <iostream>
#include <stdio.h>
#include<string.h>
using namespace std;
int main()
{
  char str1[100],str2[100],str3[100];
  scanf(" %[^\n]",&str1);
  printf("%s\n",str1);
  printf("%c\n",str1[0]);
  return 0;
}
```

- **Make a program that will show multiple strings.**

```
#include <iostream>
#include <stdio.h>
#include<string.h>
using namespace std;
int main()
{
  char str1[100],str2[100],str3[100];
  printf("Enter: ");
  scanf(" %[^\n]",&str1);
  printf("Enter: ");
  scanf(" %[^\n]",&str2);
  printf("Enter: ");
  scanf(" %[^\n]",&str3);
```

```
 printf("%s\n",str1);
  printf("%s\n",str2);
  printf("%s\n",str3);
  return 0;
}
```

- **Make a program that will show the full name of yours and your brother and sister using strings.**

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main()
{
   char str1[100],str2[100],str3[100];
   printf("Enter your full name: ");
   fflush(stdin);
   gets(str1);
   printf("Enter your brother's full name: ");
   gets(str2);
   printf("Enter your sister's full name: ");
   gets(str3);
   printf("\n");
   printf("Your name is: %s\n",str1);
   printf("Your brother name is: %s\n",str2);
   printf("Your sister name is: %s\n",str3);
   return 0;
}
```

Here,
fflush(stdin) is a library function that is used to solve the string assigning problem.

- **Make a program that will show multiple line using strings.**

```
#include <iostream>
using namespace std;
int main()
{
 char c[]="Hamim Talukder \
 Taseen";
 printf("c = %s\n",c);
 return 0;
}
```

- **Make a program that will show all the character/letter Of 'Hamim'.**

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main()
{
 char c[]="Hamim";
 printf("%c\n",c[0]);
 printf("%c\n",c[1]);
 printf("%c\n",c[2]);
 printf("%c\n",c[3]);
 printf("%c\n",c[4 ]);
 return 0;
}
```

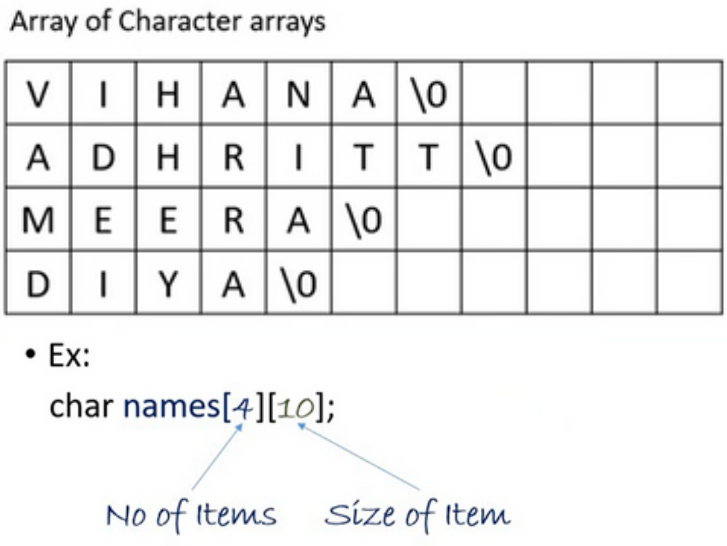
or,

```
#include <iostream>
using namespace std;
int main()
{
 char c[]="Hamim";
 int i;
 for(i=0;c[i]!='\0';i++){
 printf("%c\n",c[i]);
 }
 return 0;
}
```

Output:

```
H
a
m
i
m
```

2D character arrays of string :



- **Make a program that will show four names Hamim, Jim, Rahim, Mithu using 2D character arrays of string.**

```
#include <iostream>
#include<string.h>
using namespace std;
int main()
{
 char arr[4][10]={"Hamim","Jim","Rahim","Mithu"};
 printf("The names are :\n");
 for(int i=0;i<4;i++){
  printf("%s\n",arr[i]);
 }
 return 0;
}
```

or,

```
#include <iostream>
#include<string.h>
using namespace std;
int main()
{
  char arr[][10]={"Hamim","Jim","Rahim","Mithu"};
  printf("The names are :\n");
  for(int i=0;i<4;i++){
    printf("%s\n",arr[i]);
  }
  return 0;
}
```

Output:

```
The names are :
Hamim
Jim
Rahim
Mithu
```

- **Make a program that will show some words depending on user input using 2D character arrays of string.**

```
#include <iostream>
#include<string.h>
using namespace std;
int main()
{
 int n,i,j,k;
 printf("How many words do you want to enter : ");
 scanf("%d",&n);
 char arr[n][100];
 printf("Enter %d words :\n",n);
 for(i=0;i<n;i++){
  scanf(" %[^\n]",&arr[i]);
 }
 printf("The %d words are :\n",n);
 for(i=0;i<n;i++){
  printf("%s\n",arr[i]);
 }
 return 0;
}
```

Output:

```
How many words do you want to enter : 2
Enter 2 words :
Hridi Chowdhury
Hamim Talukder
The 2 words are :
Hridi Chowdhury
Hamim Talukder
```

- **Below is a comprehensive list of C++ Standard Library functions that are commonly used for working with character arrays (strings):**

1. C-string Functions (<cstring> header):

- **strlen(c) :** Calculates the length of a C-style string.
- **strcpy(target, source) :** Copies a C-style string to another.
- **strncpy(destination, source, num_chars) :** Copies a specified number of characters from one C-style string to another.
- **strcat(str1, str2) :** Concatenates two C-style strings by appending the second one to the first.
- **strncat(str1, str2, num_chars) :** Concatenates a specified number of characters from one C-style string to another.
- **strcmp(str1, str2) :** Compares two C-style strings lexicographically.
- **strncmp(str1, str2, num_chars) :** Compares a specified number of characters from two C-style strings.
- **strchr(str, searchChar) :** Searches for the first occurrence of a character in a C-style string.
- **strrchr(str, searchChar) :** Searches for the last occurrence of a character in a C-style string.
- **strstr(haystack, needle) :** Searches for the first occurrence of a substring in a C-style string.

- **strrev(str) :** Can reverse the string.
- **strupr(str) :** Can convert a string into upper case.
- **strlwr(str) :** Can convert a string into lower case.

Character Classification Functions (<cctype> header):

- isalpha: Checks if a character is an alphabetic letter.
- isdigit: Checks if a character is a decimal digit.
- isalnum: Checks if a character is alphanumeric (an alphabetic letter or a decimal digit).
- islower: Checks if a character is a lowercase letter.
- isupper: Checks if a character is an uppercase letter.
- isspace: Checks if a character is a whitespace character (space, tab, newline, etc.).
- iscntrl: Checks if a character is a control character.
- ispunct: Checks if a character is a punctuation character.
- isprint: Checks if a character is a printable character (including space).
- isgraph: Checks if a character has a graphical representation (excluding space).

Additional Functions:

- toupper: Converts a character to its uppercase equivalent.
- tolower: Converts a character to its lowercase equivalent.

- **strlen(c) :**
- **Make a program that will find the string length of 'Hamim Talukder'.**

```
#include <iostream>
using namespace std;
int main()
{
  char c[] = "Hamim Talukder";
  int length;
  length = strlen(c);
  printf("String length is %d\n", length);
  return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
  char c[] = "Hamim Talukder";
  int i = 0, j = 0;
  while (c[i] != '\0')
  {
    i++;
    j++;
  }
  printf("String length is %d", j);
  return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
  char c[] = "Hamim Talukder";
  int i = 0;
  while (c[i] != '\0')
  {
    i++;
  }
  printf("String length is %d", i);
  return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
 char c[]="Hamim Talukder";
 int i,j=0;
 for(i=0;c[i]!='\0';i++){
 j++;
 }
 printf("String length is %d",j);
 return 0;
}
```

Here,
strlen() is a library function that shows us the string length of any strings.

- **strcpy(target, source) :**
- **Make a program that can copy any strings.**

```
#include <iostream>
using namespace std;
int main()
{
 char source[ ] = "C Programming";
 char target[20];
 strcpy(target, source);
 printf("Source string is = %s \n", source);
 printf("Target string is = %s \n", target);
 return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
 char source[15]="Hamim Talukder";
 printf("Source string = %s\n",source);
 int i,k=0;
 for(i=0;source[i]!='\0';i++){
  k++;
 }
 printf("String length is %d\n",k);
 int j=k+1;
 char target[j];
 for(i=0;source[i]!='\0';i++){
  target[i]=source[i];
 }
```

```
 printf("Target string = %s\n",target);
 return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
 char source[15]="Hamim Talukder";
 printf("Source string = %s\n",source);
 int i,k=0;
 for(i=0;source[i]!='\0';i++){
  k++;
 }
 printf("String length is %d\n",k);
 int j=k+1;
 char target[j];
 for(i=0;source[i]!='\0';i++){
  target[i]=source[i];
 }
 printf("Target string = ");
 for(j=0;target[j]!='\0';j++){
 printf("%c",target[j]);
 }
 return 0;
}
```

Here,
strcpy() is a library function that can copy characters from a string and paste it in another string .

- **strcat(str1, str2) :**
- **Make a program that can strings concatenation .**

```
#include <iostream>
#include<string.h>
using namespace std;
int main()
{
 char str1[]="My name is ";
 char str2[]="Hamim Talukder";
 strcat(str1,str2);
 printf("%s\n",str1);
 return 0;
}
```

Here,
strcat() is a library function that can concat/add two strings.

or,

```
#include <iostream>
using namespace std;
int main()
{
 char str1[]="My name is ";
 //char str2[]="Hamim Talukder";
 strcat(str1,"Hamim Talukder");
 printf("%s\n",str1);
 return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
 char str1[]="My name is_";
 char str2[]="Hamim Talukder";
 strcat(str1,str2);
 printf("%s\n",str1);
 return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
 char str1[50]="My name is ";
 char str2[]="Hamim Talukder";
 int i,len1=0,len2=0;
 for(i=0;str1[i]!='\0';i++){
 len1++;
 }
 for(i=0;str1[i]!='\0';i++){
 str1[len1+i]=str2[i];
 }
 printf("%s\n",str1);
}
```

- **strcmp(str1, str2) :**

- **Make a program that can compare stringts.**

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
 char str1[] = "Hamim Talukder";
 char str2[] = "Hamim Talukder";
 int d = strcmp(str1, str2);
 if (d == 0)
 {
  printf("Strings are equal.");
 }
 else
  printf("Strings are not equal.");
 return 0;
}
```

Here,
strcmp() is a library function that can compare among strings.

or,

```
#include <iostream>
#include <string.h>
using namespace std;

int main()
{
   char name[50], target[30];
   cout<<"Enter your name : ";
   scanf(" %[^\n]",&name);
   cout<<"Enter your name again : ";
   scanf(" %[^\n]",&target);
   int b=0, i, j;
   for(j=0; target[j]!='\0'; j++){}
   for(i=0; name[i]!='\0'; i++){
      if(name[i]!=target[i]){
         b=1;
      }
   }

   if(b==0&&i==j){
      cout<<"Strings are equal"<<endl;
   }else{
      cout<<"Strings are not equal"<<endl;
   }
}
```

Output:

```
Enter your name : Hamim Talukder
Enter your name again : Hamim Talukder
Strings are equal
```

or,

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
 char str1[]="Hamim Talukder";
 char str2[]="Hamim Talukder";
 int i,d=0,k,s;
 k= strlen(str1);
 s= strlen(str2);
 printf("k=%d\n",k);
 printf("s=%d\n",s);
 for(i=0;str1[i]!='\0';i++){
 if(str1[i]!=str2[i]){
 d=d+1;
 }
 }
 printf("d=%d\n",d);
 if(d==0){
 printf("Strings are equal.\n");
 }
 else
 printf("Strings are not equal.\n");
 return 0;
}
```

or,

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
 char str1[]="Hamim Talukder";
 char str2[]="Hamim Talukder";
 int i,j=0,k,s;
 k= strlen(str1);
 s= strlen(str2);
 printf("%d\n",k);
 printf("%d\n",s);
 for(i=0;str1[i]!='\0';i++){
if(str1[i]==str2[i]){
 j=j+1;
 }
 }
 printf("%d\n",j);
 if(j==k&&j==s){
 printf("Strings are equal.\n");
 }
 else
 printf("Strings are not equal.\n");
 return 0;
}
```

- **strrev(str) :**

- **Make a program that will be able to reverse a string.**

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
 char str[]="Hamim Talukder";
 strrev(str);
 printf("Reverse string is : %s",str);
 return 0;
}
```

Here,
strrev() is a library function that can reverse a string

or,

```
#include <iostream>
using namespace std;
int main()
{
 char str1[] = "Hamim Talukder";
 int i, len1 = 0;
 for (i = 0; str1[i] != '\0'; i++)
 {
  len1++;
 }
```

```
printf("String length of str is %d\n",len1);
 int j=len1+1;
 char str2[j];

 for(j=0,i=len1-1;i>=0;i--,j++){
   str2[j]=str1[i];
 }
 str2!='\0';
 printf("Reverse string is : %s\n",str2);
 return 0;
}
```

or,

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
    char s[] = "I love you";
    int i;
    for (i = 0; s[i] != '\0'; i++){}
    int k = i-1;
    for (int j = 0; j < (i / 2); j++, k--)
    {
        char c = s[j];
        s[j] = s[k];
        s[k] = c;
    }
    cout << s;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
 char str[] = "Hamim Talukder";
 int i, len = 0;
 for (i = 0; str[i] != '\0'; i++)
 {
  len++;
 }
 printf("String length of str is %d\n",len);
 printf("Reverse string is : ");
 for(i=len;i>=0;i--){
  printf("%c",str[i]);
 }
 return 0;
}
```

- **Make a program that will show you palindrome strings.**

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
 char str1[30] = "madam";
 printf("string is : %s\n",str1);
 int i,j, len1 = 0;
 for (i = 0; str1[i] != '\0'; i++)
 {
 len1++;
 }
 printf("String length is %d\n",len1);
 char str2[30];
 for(j=0,i=len1-1;i>=0;i--,j++){
  str2[j]=str1[i];
 }
 printf("Reverse string is : %s\n\n",str2);
 int d=strcmp(str1,str2);
 if(d==0){
 printf("String is palindrome.\n");
 }
 else
 printf("String is not palindrome.\n");
 return 0;
}
```

or,

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
   char s[] = "madym";
   int i;
   for (i = 0; s[i] != '\0'; i++){}
   int k = i-1, b=0;
   for (int j = 0; j < (i / 2); j++, k--)
   {
      if(s[j] != s[k]){
         b=1;
      }
   }
   if(b==0){
      cout<<"String is palindrome"<<endl;
   }
   else{
      cout<<"String is not palindrome\n";
   }
}
```

Output:

String is palindrome

- **Make a program that will be able to swapping strings.**

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
  char str1[15]="Bangladesh";
  char str2[15]="India";
  char temp[15];

  printf("Before swapping :\n");
  printf("str1 = %s\n",str1);
  printf("str2 = %s\n",str2);

  strcpy(temp,str1);
  strcpy(str1,str2);
  strcpy(str2,temp);

  printf("\nAfter swapping :\n\n");
  printf("str1 = %s\n",str1);
  printf("str2 = %s\n",str2);
}
```

- **strupr(str) :**

- **Make a program that will convert a string into upper case and lower.**

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
  char str[ ]="Hamim Talukder";

  strupr(str);
  printf("Upper case of str = %s\n",str);

  return 0;
}
```

Here,
strupr() is a library function that can convert a string into upper case.

- **strlwr(str) :**

- **Make a program that will convert a string into upper case and lower.**

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
  char str[ ]="Hamim Talukder";

  strlwr(str);
  printf("Lower case of str = %s\n",str);
  return 0;
}
```

Here,
strlwr() is also a library function that can convert a string into lower case.

- **Make a program that will show number of vowels, consonants, digits, words and others.**

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
  char str[100],ch;
  int i,vowel,consonant,digit,word,other;

  printf("Enter a string : ");
  gets(str);
  i=vowel=consonant=digit=word=other=0;
 while((ch=str[i])!='\0'){
if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||ch
=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U')
 vowel++;
 else if((ch>='a' && ch<='z') || (ch>='A' && ch<='Z'))
 consonant++;
 else if(ch>='0' && ch<='9')
 digit++;
 else if(ch==' ')
 word++;
 else
 other++;
 i++;
 }
 word++;
```

```
  printf("Number of vowels : %d\n",vowel);
  printf("Number of consonants : %d\n",consonant);
  printf("Number of digits : %d\n",digit);
  printf("Number of words : %d\n",word);
  printf("Number of others : %d\n",other);
  return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
  char str[100],ch;
  int i,vowel,consonant,digit,word,other;
printf("Enter a string : ");
 gets(str);

 for(i=vowel=consonant=digit=word=other=0;
(ch=str[i])!='\0';i++){

if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||ch
=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U')
 vowel++;
 else if((ch>='a' && ch<='z') || (ch>='A' && ch<='Z'))
 consonant++;
 else if(ch>='0' && ch<='9')
 digit++;
```

```
   else if(ch==' ')
   word++;
   else
   other++;
  }
  word++;

printf("Number of vowels : %d\n",vowel);
 printf("Number of consonants : %d\n",consonant);
 printf("Number of digits : %d\n",digit);
 printf("Number of words : %d\n",word);
 printf("Number of others : %d\n",other);
 return 0;
}
```

Output:

```
Enter a string : I am Hamim 12 3
Number of vowels : 4
Number of consonants : 4
Number of digits : 3
Number of words : 5
Number of others : 0
```

- **Make a program that will show number of capital letters, small letters and digit.**

```
#include <iostream>
using namespace std;
int main()
{
  char str[100];
  int i,capital,small,digit;

  printf("Enter a string : ");
  gets(str);

  for(i=capital=small=digit=0; str[i]!='\0';i++){
   if(str[i]>=65 && str[i]<=90)
   capital++;
   else if(str[i]>=97 && str[i]<=122)
   small++;
   if(str[i]>=48 && str[i]<=57)
   digit++;
  }

  printf("Number of capital letters : %d\n",capital);
  printf("Number of small letters : %d\n",small);
  printf("Number of digits : %d\n",digit);
  return 0;
}
```

or,

```
#include <iostream>
using namespace std;
int main()
{
  char str[100];
  int i,capital,small,digit;

  printf("Enter a string : ");
  gets(str);

  i=capital=small=digit=0;
  while(str[i]!='\0'){
   if(str[i]>=65 && str[i]<=90)
   capital++;
   else if(str[i]>=97 && str[i]<=122)
   small++;
   if(str[i]>=48 && str[i]<=57)
   digit++;
   i++;
  }

  printf("Number of capital letters : %d\n",capital);
  printf("Number of small letters : %d\n",small);
  printf("Number of digits : %d\n",digit);
  return 0;
}
```

2. string class :

In C++, the std::string class is part of the Standard Template Library (STL) and is used for working with strings, which are sequences of characters. The std::string class provides a convenient and efficient way to manipulate and manage strings in C++.

- **What is the output of the following C++ program fragment:**

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
 string str1 = "Hamim";
 string str2 =" Talukder";
 string str3;
 string str4;
 str3 = str1;
 str4 = str1 + str2;
 cout<<"str1 = "<<str1<<endl;
 cout<<"str2 = "<<str2<<endl;
 cout<<"str3 = "<<str3<<endl;
 cout<<"str4 = "<<str4<<endl;
 return 0;
}
```

Output:
str1 = Hamim
str2 = Talukder
str3 = Hamim
str4 = Hamim Talukder

- **What is the output of the following C++ program fragment:**

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
 string str1 = "Hamim Talukder";
 int len = str1.size();
 cout<<"Length of str1 = "<<len<<endl;
 return 0;
}
```

Output:
Length of str1 = 14

- **What is the output of the following C++ program fragment:**

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
  string str1 = "Hello";
  char ch = 'U';
  cout<<str1[0]<<endl;
  str1[0] = ch;
  str1[1] = 'L';
  cout<<str1<<endl;
}
```

Output:
H
ULllo

- **What is the output of the following C++ program fragment:**

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
  string str1 = "Hello";
  string str2 = " World";
  string result = str1 + str2;
  cout<<"str1 : "<< str1<<", str2 : "<<str2<<",
result : "<<result<<endl;
  if(str1 == str2){
    cout<<"Equal";
  }else{
    cout<<"Not Equal";
  }
}
```

Output:

str1 : Hello, str2 : World, result : Hello World
Not Equal

- **C++ provides a rich set of functions and methods for working with strings through the Standard Template Library (STL). Below is a list of some commonly used string functions and methods:**

1. **std::string Constructor:** Creating a string using various constructors.
2. **length() or size():** Get the length (number of characters) of a string.
3. **at(index):** You can access individual characters in a string using the [] operator or at() function.
4. **substr():** Extract a substring from a string.
5. **find()**: Find the position of a substring within a string.
6. **replace():** Replace part of a string with another string.
7. **c_str():** Convert a C++ string to a C-style (null-terminated) string.

1. std::string Constructor:

- **What will be the output of the following code?**

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    // Creating strings using constructors
    string str1 = "Hello, ";
    string str2(5, 'W'); // Creates "WWWWW"
    string str3(str1 + "World!"); // Concatenates strings

    cout << str1 << str2 << str3 << endl;

    return 0;
}
```

Output:
Hello, WWWWWHello, World!

2. length() or size():

- **What will be the output of the following code?**

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  string myString = "Hello, World!";

  int length = myString.length();
  int size = myString.size();

  cout << "Length of the string: " << length << endl;
  cout << "Size of the string: " << size << endl;
  return 0;
}
```

Output:
Length of the string: 13
Size of the string: 13

3. at(index):

- **What will be the output of the following code?**

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  string myString = "Hello, World!";

  char firstChar = myString[0];
  char fifthChar = myString.at(4);

  cout << "firstChar: " << firstChar << endl;
  cout << "fifthChar: " << fifthChar << endl;
  return 0;
}
```

Output:
firstChar: H
fifthChar: o

4. substr():

- **What will be the output of the following code?**

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  string myString = "Hello, World!";

  string substring = myString.substr(7, 5);

  cout << "Substring: " << substring << endl;

  return 0;
}
```

Output:
Substring: World

5. find():

- **What will be the output of the following code?**

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  string myString = "Hello, World!";

  size_t position = myString.find("World");

  if (position != string::npos) {
    cout << "Substring found at position: " <<
position << endl;
  } else {
    cout << "Substring not found." << endl;
  }

  return 0;
}
```

Output:
Substring found at position: 7

6. replace():

- **What will be the output of the following code?**

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  string myString = "Hello, World!";

  myString.replace(7, 5, "Universe");

  cout << "Modified string: " << myString << endl;

  return 0;
}
```

Output:
Modified string: Hello, Universe!

7. c_str():

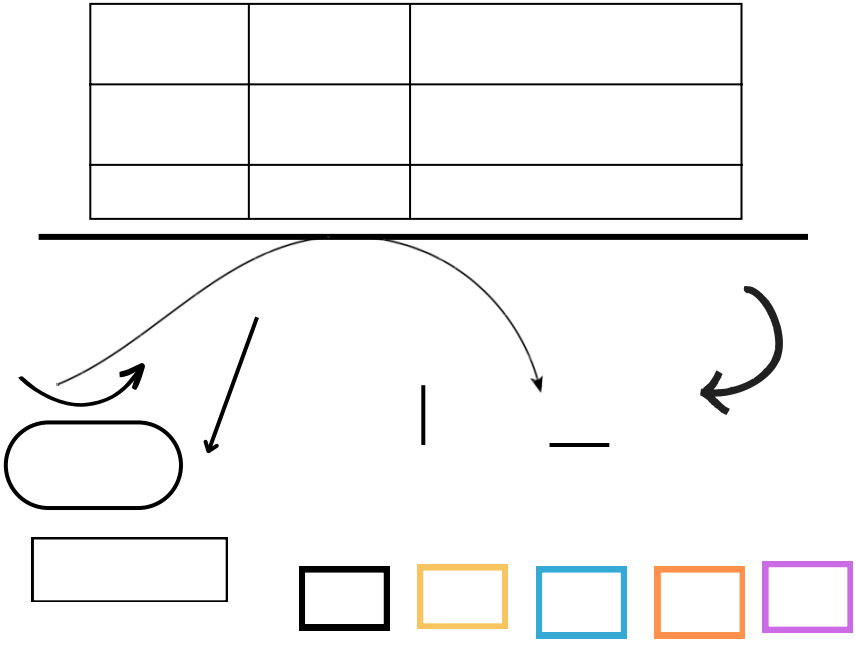
- **What will be the output of the following code?**

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  string myString = "Hello, World!";

  const char* cString = myString.c_str();

  cout << "C-style string: " << cString << endl;

  return 0;
}
```

Output:
C-style string: Hello, World!

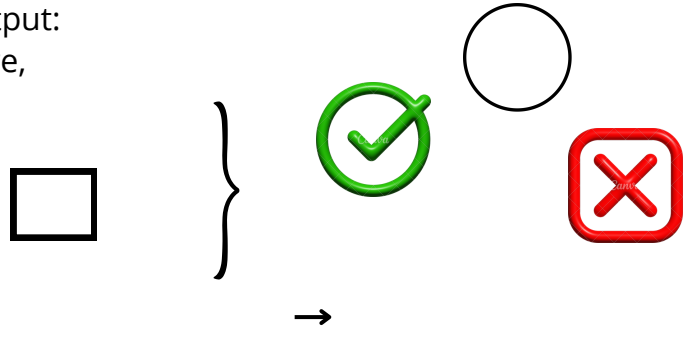
Operator Precedence and Associativity:

- **Make a program that will show String palindrome.**



Output:
Here,

Jim



- **What is the output of the following Java program fragment:**

Operator

C++
(THIRD PART)
T.I.M. HAMIM

ABC
PROKASHONI