Vector STL in C++

STL → Standard template library.

Introduction!

1. The Standard template library (STL) provides a collection of template classes and functions that offer common data structures and algorithms to make programming more efficient and convinient.

2. A Vector in C++ is a dynamic array that can grow or shrink in size, making it a versatile and efficient data structure for storing and manipulating sequence of element.

Memory Allocation (usage)

2) int arr[5]; → Static memory Allocation

int n;
cin >> n;
int *arr = new int[n]; → Dynamic memory Allocation for Array.

vector → capacity (How much capacity)
→ size

* it just double the size in the background.

Features!

1. Contiguous Memory! Elements in a vector are stored in contiguous memory locations, which makes it efficient for random Access and iteration

2. Dynamic Sizing! unlike built-in arrays in langvag (C++) which have a fixed size, vector can dynamically resize itself as elements are added or removed. This dynamic sizing is managed internally, so you don't need to worry about memory management.

3. Automatic Reallocation! when a vector reaches its capacity and you try to add more elements, it automatically reallocates memory to accomodate the new elements. This allows you to work with dynamic-sized collections without worrying about memory management.

4. Size and capacity: Vector maintains two important properties: The size, which is the number of elements present in the vector currently. and the capacity which is the number of elements the vector holds without reallocation.

(5) Array like access: Elements can be accissible like an array an using built in function ~~(at)~~; at (0) → element index

or like array v[i] → array, i → index.

Array-3

- 1D Array

int arr[5] → | 10 | 20 | 30 | 40 | 50 | 60 |

Static array

Vector: → dynamic array

Vector size insertation:

① size define → array → same like array
arr[3] = 40

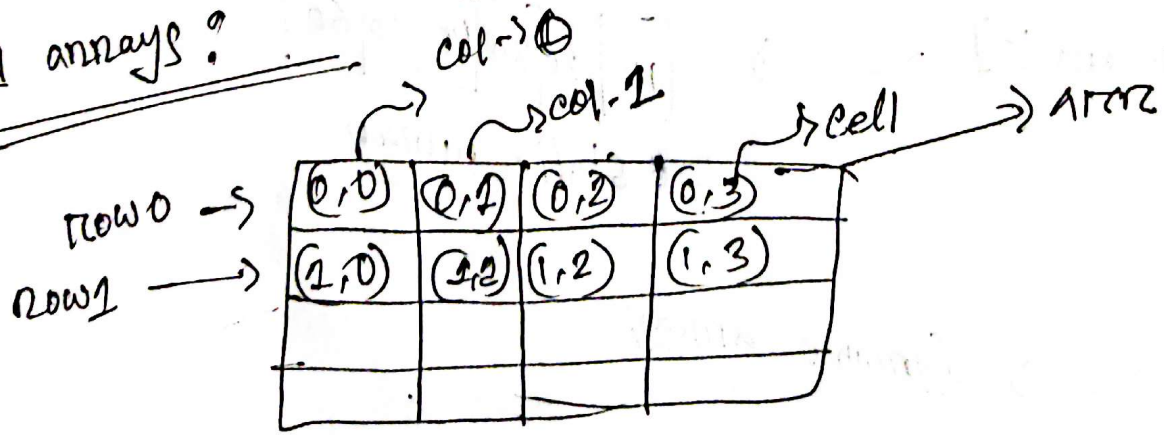
② size not → input → arr.push_back(10)

Vector traversal

```
vector <int> arr;
arr.push_back(10);
int target = 70;
```

same as array.

2d arrays:



in memory: 2d array is just a 1 d array.

(v) converting 2d array into 1d array to store in memory

⇒ formula: $(c \times i + j)$ ✓

c → total number of colums

i → row index.

j → column index.

creation of 2d array:

① declare → 1d → int arr[5]

→ 2d → int arr [4] [3] → colum

row

(vi) only in the case of static array we will get ganbage value but not in 2d array. In 2d array only 0 will be stored.

(vi) ways to traverse:

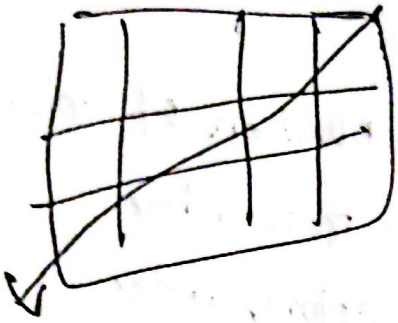
(i) row wise →

(ii) colum wise →

(iii) diagonal →

homework

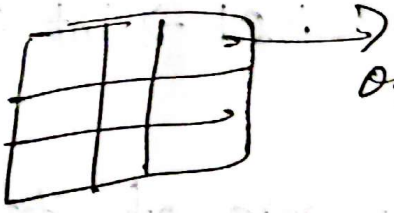
* print reverse diagonal.



* column wise input for array.

* find minimum value from 2d array.

* row wise sum



sum of all element.

* column-wise sum.

* diagonal sum.

* Transpose of matrix.