Date: 152

11/9/23 Video - Vectors

STL -> Standard Template Library

↳ collection of template classes & functions that offer common d/s & Algo to make programming efficient & convenient.

Vector -> Dynamic Array (Can say Advance version of Array)

↳ Also have Dynamic Size

↳ can grow or shrink size

ex:- int arr[5]; -> static Array (Static Memory Allocation)

```
int n;
cin>>n;
int arr[n]
```

} -> New Compiler does not accept this or Some may accept some Not

Dynamic Memory Allocation in Array

```
int n;
cin >>n;
int * arr = new int[n];
```

But have static size ↲ ↳ Dynamic memory Allocation

(n) -> i/p -> 50 ↳ can I do

| 0 — 49 | 50 | 51 |

arr[50] = 40; or arr[51] = 90;

↓ Nahi Daal Sakte

11/9/23

APCO
Date: 163

Vector → Donot tell size, just keep inserting, c++ will manage size.

Syntax →
```
#include <vector>
vector <datatype> name;
ex→ vector <int> v;
```

Insert →
```
v.push_back(1);
v.push_back(2);
v.push_back(3);
```

delete last element → `v.pop_back();`

size → `v.size()` → It's like length function java & js.

`vector <int> v;` → kuch mat karo → No Memory Allocation ← ye se allot kro memory

`v.push_back(1);` → [1] (0) ← &

`v.push_back(2);` → [1|2] (0 1)

`v.push_back(3);` → [1|2|3|1] (0 1 2 3) → empty Phegaa ! → No zero or Garbage value

Cap	Size
1	1
2	2
4	3

★ Cap → will Double if vector gets full (`v.capacity()`)

★ Size → current element in vector (`v.size()`)

11/9/23

Date: 154

Input in vector →

```
int n; cin >> n;
for (int i=0; i<n; i++){
    int d;
    cin >> d;
    v.push_back(d);
}
```

Access vector → (1) → v[i]

i → Position

(2) → v.at(i) → position

Delete all elements from vector / clear →

→ v.clear(); → size = 0

* Not capacity → 0

Initialization Types →

(1) → vector<int> a;

(2) → vector<int> v(5, 10); → with 5 duplicate values of 10

cap → 5, size → 5

10	10	10	10	10

APCO
Date: 155

11/9/23

→ can insert more element in ② Initialization

③ → vector <int> v = {1,2,3,4,5};

④ → vector <int> v {1,2,3,4,5}; → New Methods can work or cannot on old Compiler

How to copy a vector →

→ vector <int> v1;

→ vector <int> v2(v1);

Copying v1 to v2

Any change in v2 will ← Not affect v1 & vice-versa

First & Last Element →

vector <char> v | a | b | c | → v[v.size()-1] or

v[0] / v.front() / v.back()

Provided by vector

For each-Loop →

→ Automatically detect Data Type

for (auto element : a) {

Access through this Name

cout << element; ← var name

}