Kalapi

Page No.

Date

C++ STL

Lecture 1. Arrays & Vectors.

```
#include <bits/stdc++.h>
```

Struct: user defined datatype.

```
struct node
{
    int val;
    double weight;
    string name;
};
```

```
struct node
{
    int val;
    double weight;
    string name;

    node(val_, weight_, string_)
    {
        val = val_
        weight = weight_
        string = string_
    }
};
```

Constructor

Works but not a good way

```
node student1;
student1.val = 100;
student1.weight = 95.6;
student1.name = "John";
```

Best way to use struct.

```
node *student1 = new node(100, 95.6, "John");
```

or

```
node student1 = node(100, 95.6, "John");
```

Containers

Container 1) Arrays

initialize: `int arr[100];` Garbage values

~~#include~~ `array<int, 100> arr;` Garbage values

~~Max 10^6 sized int/double/char array in int main( ).~~

~~Max 10^7 " " " " globally.~~

if declared globally → All values zero.

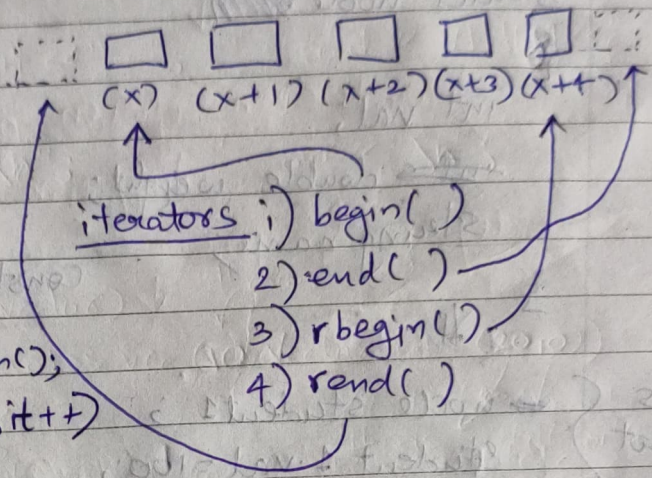
if declared in a function

Kalapi
Page No.
Date

array<int,5> arr = {1,2,3} // → {1,2,0,0,0}
" " " " = {1} // → {1,0,0,0,0}
" " " " = {0} // → {0,0,0,0,0}

int a[100]; ← same rule

i) array<int,5> arr;
Function :- arr.fill(10);
Function :- arr.at(index)



array<int,5> arr =
{2,3,9,1,0}

Loop :-
```
for(auto it = arr.begin(); it != arr.end(); it++)
{
  cout << *it << " ";
}
```

++ operator comes here;

Reverse loop :-
```
for(auto it = arr.rbegin(); it != arr.rend(); it++)
{
  cout << *it << " ";
}

for(auto it = arr.end()-1; it >= arr.begin(); it--)
{
  cout << *it << " ";
}

for(auto it = arr.rbegin(); it < arr.rend(); it++)
{
  cout << *it << " ";
}
```

Kalapi

Page No.

Date

→ can be array / string.

for-each loop :- for(auto it : arr)
{
cout << it << " ";
}

Max size of array / vector	
i)	10^6 int / double / char in main()
ii)	10^7 " " " globally
iii)	10^7 bolean in main()
iv)	10^8 " globally.

function: arr.size()
arr.front(); → arr.at(0)
arr.back(); → arr.at(arr.size()-1)

Container 2) Vectors

Declaration :- vector<int> v; // size 0

v.emplace_back() is faster than v.push_back()

function v.push_back(0);
v.push_back(1);
v → {0, 1} v.size() = 2

function v.popback();
v → {0} v.size() = 1

function:- v.clear() // → erase all elements at once → { }

Initialize :- vector<int> v (4, 0); // → {0, 0, 0, 0}
Still we can push_back into v ; 4 is just initial size

Copy entire vector (start iterator, end iterator)

i) vector<int> v1 (v.begin(), v.end()); [)
2) vector<int> v1 (v);

function:- Swap : swap(v1, v2)
clear : v.clear()

Kalapi
Page No.
Date

⇒ 2D vectors.

Declare:- `vector <vector <int >> v;`

Printing 2D vector :-

```
for(auto vctr: v)
{
  for(auto it : vctr)
  {
    cout << it << " ";
  }
  cout << "\n";
}
```

traditional way

```
for(int i=0; i<v.size(); i++)
{
  for(int j=0; j<v[i].size(); j++)
  {
    cout << v[i][j] << " ";
  }
  cout << "\n";
}
```

Declare :- (10 x 20 vector)

`vector<vector <int>> vec (10, vector<int> (20,10))`

#rows

get 10 indexes

#cols

fill with zero.

All index there is a vector of size 20, vectors filled with 0.

Declare : Array of vectors

`vector<int> arr [4];`

3D vectors

Declare :- (10 x 20 x 30 vector)

definition of 2D vector

`vector< vector <vector <int>>> vec (10, vector <vector <int>> (20, vector <int> (30,0)`