* **Github Link:** [**https://github.com/YatharthChauhan2362/DSA-Placement-Preparation**](https://github.com/YatharthChauhan2362/DSA-Placement-Preparation)
* **Link for** [**FlowChart**](https://whimsical.com/c-stl-XVxuHHof5GTWA4NXZhXQhx)

**Arrays:**

#include <iostream>

#include<array>

using namespace std;

int main() {

int basic[3] ={1,2,3};

array<int,4> a = {1,2,3,4};

int size = a.size();

for(int i=0;i<size;i++ ){

cout<<a[i]<<endl;

}

cout<<"Element at 2nd Index-> "<<a.at(2)<<endl;

cout<<"Empty or not-> "<<a.empty()<<endl;

cout<<"First Element-> "<<a.front()<<endl;

cout<<"last Element-> "<<a.back()<<endl;

}

**Vector:**

#include <iostream>

#include<vector>

using namespace std;

int main() {

vector<int> v;

vector<int> a(5,1);

vector<int> last(a);

cout<<"print last"<<endl;

for(int i:last) {

cout<<i<<" ";

}cout<<endl;

cout<<"Capacity-> "<<v.capacity()<<endl;

v.push\_back(1);

cout<<"Capacity-> "<<v.capacity()<<endl;

v.push\_back(2);

cout<<"Capacity-> "<<v.capacity()<<endl;

v.push\_back(3);

cout<<"Capacity-> "<<v.capacity()<<endl;

cout<<"Size-> "<<v.size()<<endl;

cout<<"Elemetn at 2nd Index" <<v.at(2)<<endl;

cout<<"front " <<v.front()<<endl;

cout<<"back " <<v.back()<<endl;

cout<<"before pop"<<endl;

for(int i:v) {

cout<<i<<" ";

}cout<<endl;

v.pop\_back();

cout<<"after pop"<<endl;

for(int i:v) {

cout<<i<<" ";

}

cout<<"before clear size "<<v.size()<<endl;

v.clear();

cout<<"after clear size "<<v.size()<<endl;

}

**Deque:**

#include <iostream>

#include<deque>

using namespace std;

int main() {

deque<int> d;

d.push\_back(1);

d.push\_front(2);

//d.pop\_front();

cout<<endl;

cout<<"Print First INdex Element-> "<<d.at(1)<<endl;

cout<<"front "<<d.front()<<endl;

cout<<"back "<<d.back()<<endl;

cout<<"Empty or not" <<d.empty()<<endl;

cout<<"before erase" <<d.size()<<endl;

d.erase(d.begin(),d.begin()+1);

cout<<"after erase" <<d.size()<<endl;

for(int i:d){

cout<<i<<endl;

}

}

**List:**

#include <iostream>

#include<list>

using namespace std;

int main() {

list<int> l;

list<int> n(5,100);

cout<<"Printing n"<<endl;

for(int i:n) {

cout<<i<<" ";

}cout<<endl;

l.push\_back(1);

l.push\_front(2);

for(int i:l) {

cout<<i<<" ";

}

cout<<endl;

l.erase(l.begin());

cout<<"after erase"<<endl;

for(int i:l) {

cout<<i<<" ";

}

cout<<"size of list"<<l.size()<<endl;

}

**Stack:**

#include <iostream>

#include<stack>

using namespace std;

int main() {

stack<string> s;

s.push("love");

s.push("babbar");

s.push("Kumar");

cout<<"Top Element-> "<<s.top()<<endl;

s.pop();

cout<<"Top Element-> "<<s.top()<<endl;

cout<<"size of stack"<<s.size()<<endl;

cout<<"Empty or not "<<s.empty()<<endl;

}

**Queue:**

#include <iostream>

#include<queue>

using namespace std;

int main() {

queue<string> q;

q.push("love");

q.push("Babbar");

q.push("Kumar");

cout<<"Size before pop" <<q.size()<<endl;

cout<<"First Element "<<q.front()<<endl;

q.pop();

cout<<"First Element "<<q.front()<<endl;

cout<<"Size after pop" <<q.size()<<endl;

}

**Priority\_queue:**

#include <iostream>

#include<queue>

using namespace std;

int main() {

//max heap

priority\_queue<int> maxi;

//min - heap

priority\_queue<int,vector<int> , greater<int> > mini;

maxi.push(1);

maxi.push(3);

maxi.push(2);

maxi.push(0);

cout<<"size-> "<<maxi.size()<<endl;

int n = maxi.size();

for(int i=0;i<n;i++) {

cout<<maxi.top()<<" ";

maxi.pop();

}cout<<endl;

mini.push(5);

mini.push(1);

mini.push(0);

mini.push(4);

mini.push(3);

int m = mini.size();

for(int i=0;i<m;i++) {

cout<<mini.top()<<" ";

mini.pop();

}cout<<endl;

cout<<"khaali h kya bhai ?? -> "<<mini.empty()<<endl;

}

**Set:**

#include <iostream>

#include<set>

using namespace std;

int main() {

set<int> s;

s.insert(5);

s.insert(5);

s.insert(5);

s.insert(1);

s.insert(6);

s.insert(6);

s.insert(0);

s.insert(0);

s.insert(0);

for(auto i : s) {

cout<<i<<endl;

}cout<<endl;

set<int>::iterator it = s.begin();

it++;

s.erase(it);

for(auto i : s) {

cout<<i<<endl;

}

cout<<endl;

cout<<"-5 is present or not -> "<<s.count(-5)<<endl;

set<int>::iterator itr = s.find(5);

for(auto it=itr;it!=s.end();it++) {

cout<<\*it<<" ";

}cout<<endl;

}

**Map:**

#include <iostream>

#include<map>

using namespace std;

int main() {

map<int,string> m;

m[1]= "babbar";

m[13]="kumar";

m[2]="love";

m.insert( {5,"bheem"});

cout<<"before erase"<<endl;

for(auto i:m) {

cout<<i.first<<" "<<i.second<<endl;

}

cout<<"finding -13 -> " <<m.count(-13)<<endl;

// m.erase(13);

cout<<"after erase"<<endl;

for(auto i:m) {

cout<<i.first<<" "<<i.second<<endl;

}cout<<endl<<endl;

auto it = m.find(5);

for(auto i=it;i!=m.end();i++) {

cout<<(\*i).first<<endl;

}

}

**Algo:**

#include <iostream>

#include<algorithm>

#include<vector>

using namespace std;

int main() {

vector<int> v;

v.push\_back(1);

v.push\_back(3);

v.push\_back(6);

v.push\_back(7);

cout<<"Finding 6-> "<<binary\_search(v.begin(),v.end(),6)<<endl;

cout<<"lower bound-> "<<lower\_bound(v.begin(),v.end(),6)-v.begin()<<endl;

cout<<"Uppper bound-> "<<upper\_bound(v.begin(),v.end(),4)-v.begin()<<endl;

int a =3;

int b =5;

cout<<"max -> "<<max(a,b);

cout<<"min -> "<<min(a,b);

swap(a,b);

cout<<endl<<"a-> "<<a<<endl;

string abcd = "abcd";

reverse(abcd.begin(),abcd.end());

cout<<"string-> "<<abcd<<endl;

rotate(v.begin(),v.begin()+1,v.end());

cout<<"after rotate"<<endl;

for(int i:v){

cout<<i<<" ";

}

sort(v.begin(),v.end());

cout<<"after sorting"<<endl;

for(int i:v){

cout<<i<<" ";

}

}