**ARRAY**

* Size of array

s = sizeof(arr)/sizeof(arr[0]);

* Sorting an array (ascending)

sort(array\_name, array\_name+sizeOfArray);

* Sorting an array (descending)

bool comp(int x, int y){

if(x>y) return true;

return false;

}

int main() {

int arr[10]= {1,2,5,3,4,2,8,34,1,2};

sort(arr, arr+10, comp);

}

* Memset

memset(array\_name, 0 , sizeof(array\_name)); // to initialize an array

* First and last elemnt

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

cout<<arr.front(); //1

cout<<arr.back(); //10

arr.fill(20); // {20,20,20,20,20,20,20,20,20,20}

**DEQUE**

* Push and pop from front and back
* Push in specific position

deque <int> d;

degue <int> a(10); //size of deque is 10

push\_front(int x);

push\_back(int x);

pop\_front();

pop\_back();

d.front();

d.back();

**MAP**

* Can count the number of occurance

map <string , int> mp;

mp["Canada"] =5;

mp["BD"]++;

map <string, int> ::iterator itr;

for(itr = mp.begin(); itr!=mp.end(); itr++){

cout<<itr -> first<<"\t"<<itr->second<<endl; //first ----key, second ----value

}

if (mp[key]==0)

map <string, int, greater<string>> mp;

**PAIR**

pair <dataType1, dataType2> p;

Example: pair <int , pair<int, int>> p;

int a = p.second.first;

Example: pair <int, int> p[100];

p.first =1;

int arr[10];

arr[0] = make\_pair(1,2);

sort(p, p+size); //ascendeing sort

sort(p, p+size, comp); //descending sort

**PRIORITY QUEUE**

* Insert in sorted order

// initialize

priority\_queue <int> pq; // ascending

priority\_queue <int, vector <int>, greater <int>> descPq; // decending

// same as queue

**QUEUE(FIFO)**

* First in first out

queue <int> q;

q.push(4);

q.pop();

int a = q.front();

int b = q.back();

int sizeOfQueue = q.size();

bool flag = q.empty();

queue1 = {1, 2, 3, 4}

queue2 = {5, 6, 7, 8}

queue1.swap(queue2);

queue1.emplace(6);

queue1 = [ 6, 1, 2, 3, 4 ]

**SET**

* Auto sorted in ascending sort
* Takes only the unique values

set <string> st;

set <string> ::iterator itr;

for(itr = st.begin(); itr!=st.end(); itr++){

cout<<"Element "<<\*itr<<endl;

}

* Sort

struct cmp {

bool operator() (const int& l, const int& r) {

return l < r;

}

};

int main() {

set <int> st1; // normal ascending sort

set <int, cmp> st2; // descending sort

st1.insert(4);

st1.insert(1);

st1.insert(67);

st1.insert(4);

st2.insert(4);

st2.insert(1);

st2.insert(67);

st2.insert(4);

set <int>::iterator it;

for (it = st1.begin(); it != st1.end(); it++) cout << \*it << endl;

}

**STACK(LIFO)**

* Last in first out

stack <int> s;

s.push(4);

s.pop();

int a = s.top();

bool flag = s.empty();

int sizeOfStack = s.size();

showstack(s);

void showstack(stack <int> s) {

while (!s.empty()){

cout << s.top();

s.pop();

}

cout << '\n';

}

mystack1 = {7, 5, 3, 1}

mystack2 = {8, 6, 4, 2}

mystack1.swap(mystack2);

**STRING**

* index of the first character of a word

string s = "Somewhere something incredible!";

int firstSome = s.find("some"); //it will be 0

int lastSome = s.rfind("some"); // it will be 10

* substring between two index

string st = s.substr(2,9); // it will be "mewhere"

* Minimum and maximum string between two substring

string mini = min(s.substr(0,9), s.substr(11,19));

string maxi = max(s.substr(0,9), s.substr(11,19));

* Upper to lower and vice versa (for whole string and for a character in the string)

string s = "FarIa";

transform(s.begin(), s.end(), s.begin(), ::tolower); //faria

transform(s.begin(), s.end(), s.begin(), ::tolower); //FARIA

s[2]= tolower(s[2]);

cout<<s; //FArIA

for(int i=0; i<s.length(); i++){

if(s[i]>='a' && s[i]<='z'){

s[i] =s[i]-32;

}else if(s[i]>='A' && s[i]<='Z'){

s[i] =s[i]+32;

}

}

* Sorting characters of a string in lexicographic order

sort(s.begin(), s.end()); //AAFIr

* Comparing two string

string str1 = "Faizun";

string str2 = "Faria";

if(str1.compare(str2)<0){

cout<<str1<<" comes first"<<endl;

}

* Length of string

int l = str1.length();

* character of string comparison

if (s[i]=='a')

* Reverse the order of characters in a string

s = reverse( s.begin() , s.end() );// bad will be dab

* comparing two string of integer

int a= "1289";

int b = "514";

if(a.length()<b.length()){

swap(a,b);

}

while(b.length()<a.length()){

b="0"+b;

}

* Check if a character is alphabet

bool isLetter(char c1){

return ( ( (c1>='a') && (c1<='z') ) || ( (c1>='A') && (c1<='Z') ) );

}

**Vector**

* Can get value of a particular index
* Can insert in the back
* Not sorted
* No size

vector <data\_type> v;

int sizeOfVector = v.size();

v.push\_back(12); //Insert

int first = v.front();

int last = v.back();

int fifthElement = v[4]; //Vector index start from zero

v.empty()? cout<<"vector is empty"<<endl : cout<<"vector is not empty"<<endl;

v.pop\_back(); // delete last value of the vector

* Swap two vector

swap(v1, v2);

* Sort

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

sort(v.rbegin(), v.rend()); //descending

* Remove

v.clear() //vector clear

v.erase(v.begin(), v.end()); //remove all the elements

\* Index wise element removing

v = {1,2,3,4,5,6,7}

int iterator1 = 2;

int iterator2 = 4;

v.erase(iterator1);

v = {1,2,4,5,6,7}

v.erase(2,4); // Erase from index 2 to 4

v = {1,2,7}

* Printing elements in both ascending and descending order

vector <int> v;

vector <int> :: iterator it1;

vector<int> :: reverse\_iterator it2;

for(it1 = v.begin(); it1!=v.end(); it1++) cout<< \*it1<<" ";

cout<<"\n";

for(it2 = v.rbegin(); it2!=v.rend(); it2++) cout<< \*it2<<" ";

cout<<"\n";

* assigning same value multiple times

vector<int> vec;

v1.push\_back(32);

v.assign(3, 20); // Clear the vector and the insert element 20 into vector 3 times

v.push\_back(30); // v={20,20,20,30}

* Copying some elements of one vector to another

vector<int> vec2;

vector <int> ::iterartor it;

it = vec.begin();

vec2.assign(it+1, vec.end());

for(int i=0; i<vec2.size(); i++){

cout<<vec2[i]<<" ";

}

* Inserting into any index

vector <int> vec3(3, 10);

vector<int > ::iterator it;

it = vec3.begin();

it = vec3.insert(it, 20); //inserting into first index

for(int i=0; i<vec3.size(); i++){

cout<<vec3[i]<<" ";

}

* No duplicate (Vector must be sorted beforehand)

v.erase( unique( v.begin() , v.end() ) , v.end() );

* Without sorting finding the minimum element

min\_element( v.begin() , v.end() );

**VECTOR PAIR**

vector <int,pair<int,int>> vp;

cout<<v[i].first;

vector <pair <int, int>> cans;

cans.push\_back(make\_pair(a,b)); //a and b are integer

**CONVERT (INT, STRING, CHARACTER, DOUBLE, )**

* Character array to string

char a[] = "Faria";

string str = std:: string(a);

* Set to vector

vector <int> v;

set <int> s;

v.assign( s.begin() , s.end() );