std::vector<int> myvector;

myvector.push\_back (myint);

while (!myvector.empty())

{

sum+=myvector.back();

myvector.pop\_back();

}

// erase the 6th element

myvector.erase (myvector.begin()+5);

std::vector<int> myvector (3,100);

std::vector<int>::iterator it;

it = myvector.begin();

it = myvector.insert ( it , 200 );

myvector.insert (it,2,300);

// "it" no longer valid, get a new one:

it = myvector.begin();

std::vector<int> anothervector (2,400);

myvector.insert (it+2,anothervector.begin(),anothervector.end());

// sort algorithm example

#include <iostream> // std::cout

#include <algorithm> // std::sort

#include <vector> // std::vector

bool myfunction (int i,int j) { return (i<j); }

struct myclass {

bool operator() (int i,int j) { return (i<j);}

} myobject;

int main () {

int myints[] = {32,71,12,45,26,80,53,33};

std::vector<int> myvector (myints, myints+8); // 32 71 12 45 26 80 53 33

// using default comparison (operator <):

std::sort (myvector.begin(), myvector.begin()+4); //(12 32 45 71)26 80 53 33

// using function as comp

std::sort (myvector.begin()+4, myvector.end(), myfunction); // 12 32 45 71(26 33 53 80)

// using object as comp

std::sort (myvector.begin(), myvector.end(), myobject); //(12 26 32 33 45 53 71 80)

// print out content:

std::cout << "myvector contains:";

for (std::vector<int>::iterator it=myvector.begin(); it!=myvector.end(); ++it)

std::cout << ' ' << \*it;

std::cout << '\n';

return 0;

**search map by key**

#include <unordered\_map>

int main (){

std::unordered\_map<std::string,double> mymap = {

{"mom",5.4},

{"dad",6.1},

{"bro",5.9} };

std::string input;

std::cout << "who? ";

getline (std::cin,input); **std::unordered\_map<std::string,double>::const\_iterator** got = mymap.**find** (input);

if ( got == mymap.end() )

std::cout << "not found";

else

std::cout << got->first << " is " << got→second;

for (auto& x: {"mom","dad","bro","sis"}) {

if (mymap.count(x)>0)

std::cout << "mymap has " << x << std::endl;

else

std::cout << "mymap has no " << x << std::endl;

}

**search map by value**

std::unordered\_map<std::string,double>::const\_iterator it;

for(it = mymap.begin(); it!= mymap.end(); ++it ){

if (it->second == 6.1)

std::cout<< it->first << "=" << it->second << std::endl;

}

std::unordered\_map<std::string,std::string> mymap;

mymap["Bakery"]="Barbara"; // new element inserted

mymap["Seafood"]="Lisa"; // new element inserted

mymap["Produce"]="John"; // new element inserted

std::string name = mymap["Bakery"]; // existing element accessed (read)

mymap["Seafood"] = name; // existing element accessed (written)

mymap["Bakery"] = mymap["Produce"]; // existing elements accessed (read/written)

name = mymap["Deli"]; // non-existing element: new element "Deli" inserted!

mymap["Produce"] = mymap["Gifts"];

Seafood: Barbara

Deli:

Bakery: John

Gifts:

Produce:

#include <iostream>

#include <map>

#include <vector>

#include <algorithm> // for sort function

using namespace std;

// utility comparator function to pass to the sort() module

bool sortByVal(const pair<string, int> &a,

const pair<string, int> &b)

{

return (a.second < b.second);

}

int main()

{

// create the map

map<string, int> mymap = {

{"coconut", 10}, {"apple", 5}, {"peach", 30}, {"mango", 8}

};

cout << "The map, sorted by keys, is: " << endl;

map<string, int> :: iterator it;

for (it=mymap.begin(); it!=mymap.end(); it++)

{

cout << it->first << ": " << it->second << endl;

}

cout << endl;

// create a empty vector of pairs

vector<pair<string, int>> vec;

// copy key-value pairs from the map to the vector

map<string, int> :: iterator it2;

for (it2=mymap.begin(); it2!=mymap.end(); it2++)

{

vec.push\_back(make\_pair(it2->first, it2->second));

}

// // sort the vector by increasing order of its pair's second value

sort(vec.begin(), vec.end(), sortByVal);

// print the vector

cout << "The map, sorted by value is: " << endl;

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

{

cout << vec[i].first << ": " << vec[i].second << endl;

}

return 0;

}

int two\_d[10][20];

class Solution {

private:

void dfs(vector<vector<char>>& grid, int r, int c){

int nr = grid.size();

int nc = grid[0].size();

grid[r][c] = '0';

if (r - 1 >= 0 && grid[r-1][c] == '1') dfs(grid, r - 1, c);

if (r + 1 < nr && grid[r+1][c] == '1') dfs(grid, r + 1, c);

if (c - 1 >= 0 && grid[r][c-1] == '1') dfs(grid, r, c - 1);

if (c + 1 < nc && grid[r][c+1] == '1') dfs(grid, r, c + 1);

};

public:

int numIslands(vector<vector<char>>& grid) {

int nr = grid.size();

if(!nr) return 0;

int nc = grid[0].size();

int num\_islands = 0;

for(int r = 0; r < nr; ++r){

for (int c = 0; c < nc; ++c){

if(grid[r][c]=='1'){

++num\_islands;

dfs(grid, r, c);

}

}

}

return num\_islands;

}

};

**delete map item by iterator, key, range**

// erase examples:

mymap.erase ( mymap.begin() ); // erasing by iterator

mymap.erase ("France"); // erasing by key

mymap.erase ( mymap.find("China"), mymap.end() ); // erasing by range

// Take any two unordered\_map

unordered\_map<int, int> ump1, ump2;

// Inserting values

ump1[1] = 2;

ump1[3] = 4;

ump1[5] = 6;

ump1[7] = 8;

// Print the size of container

cout << "ump1 size = " << ump1.size() << endl;

cout << "ump2 size = " << ump2.size() << endl;

// Running the function for ump1

if (ump1.empty())

cout << "True\n";

else

cout << "False\n";

// Running the function for ump2

if (ump2.empty())

cout << "True\n";

else

cout << "False\n";