SORTING VECTOR OF PAIRS

Sorting Vector of Pairs in C++ | Set 1 (Sort by first and second)

What is Vector of Pairs?  
A [pair](http://quiz.geeksforgeeks.org/pair-simple-containers-the-c-standard-template-library-stl/) is a container which stores two values mapped to each other, and a [vector](http://quiz.geeksforgeeks.org/vector-sequence-containers-the-c-standard-template-library-stl-set-1/) containing multiple number of such pairs is called a vector of pairs.

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| // C++ program to demonstrate vector of pairs  #include<bits/stdc++.h>  using namespace std;    int main()  {      //declaring vector of pairs      vector< pair <int,int> > vect;        // initialising 1st and 2nd element of      // pairs with array values      int arr[] = {10, 20, 5, 40 };      int arr1[] = {30, 60, 20, 50};      int n = sizeof(arr)/sizeof(arr[0]);        // Entering values in vector of pairs      for (int i=0; i<n; i++)          vect.push\_back( make\_pair(arr[i],arr1[i]) );        // Printing the vector      for (int i=0; i<n; i++)      {          // "first" and "second" are used to access          // 1st and 2nd element of pair respectively          cout << vect[i].first << " "               << vect[i].second << endl;      }        return 0;  } |

Output:

10 30

20 60

5 20

40 50

Case 1 : Sorting the vector elements on the basis of first element of pairs in ascending order.  
This type of sorting can be achieved using simple “ sort() ” function. By default the sort function sorts the vector elements on basis of first element of pairs.

|  |
| --- |
| // C++ program to demonstrate sorting in  // vector of pair according to 1st element  // of pair  #include<bits/stdc++.h>  using namespace std;    int main()  {      // Declaring vector of pairs      vector< pair <int,int> > vect;        // Initializing 1st and 2nd element of      // pairs with array values      int arr[] = {10, 20, 5, 40 };      int arr1[] = {30, 60, 20, 50};      int n = sizeof(arr)/sizeof(arr[0]);        // Entering values in vector of pairs      for (int i=0; i<n; i++)          vect.push\_back( make\_pair(arr[i],arr1[i]) );        // Printing the original vector(before sort())      cout << "The vector before sort operation is:\n" ;      for (int i=0; i<n; i++)      {          // "first" and "second" are used to access          // 1st and 2nd element of pair respectively          cout << vect[i].first << " "               << vect[i].second << endl;        }        // Using simple sort() function to sort      sort(vect.begin(), vect.end());         // Printing the sorted vector(after using sort())      cout << "The vector after sort operation is:\n" ;      for (int i=0; i<n; i++)      {          // "first" and "second" are used to access          // 1st and 2nd element of pair respectively          cout << vect[i].first << " "               << vect[i].second << endl;      }        return 0;  } |

Output:

The vector before applying sort operation is:

10 30

20 60

5 20

40 50

The vector after applying sort operation is:

5 20

10 30

20 60

40 50

Case 2 : Sorting the vector elements on the basis of second element of pairs in ascending order.  
There are instances when we require to sort the elements of vector on the basis of second elements of pair. For that, we modify the sort() function and we pass a third argument, a call to an user defined explicit function in the sort() function.

|  |
| --- |
| // C++ program to demonstrate sorting in vector  // of pair according to 2nd element of pair  #include<bits/stdc++.h>  using namespace std;    // Driver function to sort the vector elements  // by second element of pairs  bool sortbysec(const pair<int,int> &a,                const pair<int,int> &b)  {      return (a.second < b.second);  }    int main()  {      // declaring vector of pairs      vector< pair <int, int> > vect;        // Initialising 1st and 2nd element of pairs      // with array values      int arr[] = {10, 20, 5, 40 };      int arr1[] = {30, 60, 20, 50};      int n = sizeof(arr)/sizeof(arr[0]);        // Entering values in vector of pairs      for (int i=0; i<n; i++)          vect.push\_back( make\_pair(arr[i],arr1[i]) );        // Printing the original vector(before sort())      cout << "The vector before sort operation is:\n" ;      for (int i=0; i<n; i++)      {          // "first" and "second" are used to access          // 1st and 2nd element of pair respectively          cout << vect[i].first << " "               << vect[i].second << endl;        }        // Using sort() function to sort by 2nd element      // of pair      sort(vect.begin(), vect.end(), sortbysec);        // Printing the sorted vector(after using sort())      cout << "The vector after sort operation is:\n" ;      for (int i=0; i<n; i++)      {          // "first" and "second" are used to access          // 1st and 2nd element of pair respectively          cout << vect[i].first << " "               << vect[i].second << endl;      }      return 0;  } |

SORTING MAPS

A.USING VECTORS

#include <iostream>

#include <map>

#include <vector>

#include <algorithm>

typedef std::pair<std::string,int> pair;

int main()

{// input map

std::map<std::string,int> map = {

{"two", 2}, {"one", 1}, {"four", 4}, {"three", 3}

};

// create a empty vector of pairs

std::vector<pair> vec;

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

std::copy(map.begin(),

map.end(),

std::back\_inserter<std::vector<pair>>(vec));

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

// if second value are equal, order by the pair's first value

std::sort(vec.begin(), vec.end(),

[](const pair& l, const pair& r) {

if (l.second != r.second)

return l.second < r.second;

return l.first < r.first;

});

// print the vector

for (auto const &pair: vec) {

std::cout << '{' << pair.first << "," << pair.second << '}' << '\n';

}

return 0;

}

B.USING SET

struct comp

{

template<typename T>

bool operator()(const T& l, const T& r) const

{

if (l.second != r.second)

return l.second < r.second;

return l.first < r.first;

}

};

int main()

{

// input map

std::map<std::string,int> map = {

{"two", 2}, {"one", 1}, {"four", 4}, {"three", 3}

};

// create a empty vector of pairs

std::set<std::pair<std::string,int>, comp> set(map.begin(), map.end());

// print the vector

for (auto const &pair: set) {

std::cout << '{' << pair.first << "," << pair.second << '}' << '\n';

}

return 0;

}

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

If u need to print value from specific key you can use find member function:

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 5 6 7 | std::map<std::string, int>::iterator it;  if((it = mymap.find("32/2")) != mymap.end())  std::cout << it->first<<" =>"<< it->second << '\n';  //or  auto it = mymap.find("32/2");  if (it != mymap.end())  std::cout << it->first<<" =>"<< it->second << '\n'; |  |

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**NOTE: There is a found() function for SETS.**

**vector.assign() vector.fill()**

**vector.clear() vector.resize()**