Pair in C++ Standard Template Library (STL)

The pair container is a simple container defined in **<utility>** header consisting of two data elements or objects.

* The first element is referenced as ‘first’ and the second element as ‘second’ and the order is fixed (first, second).
* Pair is used to combine together two values which may be different in type. Pair provides a way to store two heterogeneous objects as a single unit.
* Pair can be assigned, copied and compared. The array of objects allocated in a map or hash\_map are of type ‘pair’ by default in which all the ‘first’ elements are unique keys associated with their ‘second’ value objects.
* To access the elements, we use variable name followed by dot operator followed by the keyword first or second.

**Syntax :**

pair (data\_type1, data\_type2) Pair\_name;

|  |
| --- |
| //CPP program to illustrate pair STL  #include <iostream>  #include <utility>  using namespace std;    int main()  {      pair <int, char> PAIR1 ;        PAIR1.first = 100;      PAIR1.second = 'G' ;        cout << PAIR1.first << " " ;      cout << PAIR1.second << endl ;        return 0;  } |

Output:

100 G

**Initializing a pair**

We can also initialize a pair.  
Syntax :

pair (data\_type1, data\_type2) Pair\_name (value1, value2) ;

Different ways to initialize pair:

pair g1; //default

pair g2(1, 'a'); //initialized, different data type

pair g3(1, 10); //initialized, same data type

pair g4(g3); //copy of g3

Another way to initialize a pair is by using the make\_pair() function.

g2 = make\_pair(1, 'a');

|  |
| --- |
| //CPP program to illustrate Initializing of pair STL  #include <iostream>  #include <utility>  using namespace std;    int main()  {      pair <string,double> PAIR2 ("GeeksForGeeks", 1.23);        cout << PAIR2.first << " " ;      cout << PAIR2.second << endl ;    return 0;  } |

Output:

GeeksForGeeks 1.23

**Note:** If not initialized, the first value of the pair gets automatically initialized.

|  |
| --- |
| //CPP program to illustrate auto-initializing of pair STL  #include <iostream>  #include <utility>    using namespace std;    int main()  {      pair <int, double> PAIR1 ;      pair <string, char> PAIR2 ;        cout << PAIR1.first ;  //it is initialised to 0      cout << PAIR1.second ; //it is initialised to 0        cout << " ";        cout << PAIR2.first ;  //it prints nothing i.e NULL      cout << PAIR2.second ; //it prints nothing i.e NULL        return 0;  } |

Output:

00

**Member Functions**

1. **make\_pair()** : This template function allows to create a value pair without writing the types explicitly.  
   Syntax :

Pair\_name = make\_pair (value1,value2);

|  |
| --- |
| #include <iostream>  #include <utility>  using namespace std;    int main()  {      pair <int, char> PAIR1 ;      pair <string, double> PAIR2 ("GeeksForGeeks", 1.23) ;      pair <string, double> PAIR3 ;        PAIR1.first = 100;      PAIR1.second = 'G' ;        PAIR3 = make\_pair ("GeeksForGeeks is Best",4.56);        cout << PAIR1.first << " " ;      cout << PAIR1.second << endl ;        cout << PAIR2.first << " " ;      cout << PAIR2.second << endl ;        cout << PAIR3.first << " " ;      cout << PAIR3.second << endl ;        return 0;  } |

Output:

100 G

GeeksForGeeks 1.23

GeeksForGeeks is Best 4.56

1. **operators(=, ==, !=, >=, <=) :** We can use operators with pairs as well.
   * **using equal(=) :** It assigns new object for a pair object.  
     Syntax :

pair& operator= (const pair& pr);

This Assigns pr as the new content for the pair object. The first value is assigned the first value of pr and the second value is assigned the second value of pr .

* + **Comparison (==) operator with pair :** For given two pairs say pair1 and pair2, the comparison operator compares the first value and second value of those two pairs i.e. if pair1.first is equal to pair2.first or not AND if pair1.second is equal to pair2.second or not .
  + **Not equal (!=) operator with pair :** For given two pairs say pair1 and pair2, the != operator compares the first values of those two pairs i.e. if pair1.first is equal to pair2.first or not, if they are equal then it checks the second values of both.
  + **Logical( >=, <= )operators with pair :** For given two pairs say pair1 and pair2, the =, >, can be used with pairs as well. It returns 0 or 1 by only comparing the first value of the pair.

.

|  |
| --- |
| //CPP code to illustrate operators in pair  #include <iostream>  #include<utility>  using namespace std;    int main()  {      pair<int, int>pair1 = make\_pair(1, 12);      pair<int, int>pair2 = make\_pair(9, 12);          cout << (pair1 == pair2) << endl;      cout << (pair1 != pair2) << endl;      cout << (pair1 >= pair2) << endl;      cout << (pair1 <= pair2) << endl;      cout << (pair1 > pair2) << endl;      cout << (pair1 < pair2) << endl;        return 0;  } |

Output:

0

1

0

1

0

1

1. **swap :**This function swaps the contents of one pair object with the contents of another pair object. The pairs must be of same type.  
   Syntax :

pair1.swap(pair2) ;

For two given pairs say pair1 and pair2 of same type, swap function will swap the pair1.first with pair2.first and pair1.second with pair2.second.

|  |
| --- |
| #include <iostream>  #include<utility>    using namespace std;    int main()  {      pair<char, int>pair1 = make\_pair('A', 1);      pair<char, int>pair2 = make\_pair('B', 2);        cout << "Before swapping:\n " ;      cout << "Contents of pair1 = " << pair1.first << " " << pair1.second ;      cout << "Contents of pair2 = " << pair2.first << " " << pair2.second ;      pair1.swap(pair2);        cout << "\nAfter swapping:\n ";      cout << "Contents of pair1 = " << pair1.first << " " << pair1.second ;      cout << "Contents of pair2 = " << pair2.first << " " << pair2.second ;        return 0;  } |

Output:

Before swapping:

Contents of pair1 = (A, 1)

Contents of pair2 = (B, 2)

After swapping:

Contents of pair1 = (B, 2)

Contents of pair2 = (A, 1)

|  |
| --- |
| //CPP program to illustrate pair in STL  #include <iostream>  #include <utility>  #include <string>  using namespace std;    int main()  {      pair <string, int> g1;      pair <string, int> g2("Quiz", 3);      pair <string, int> g3(g2);      pair <int, int> g4(5, 10);        g1 = make\_pair(string("Geeks"), 1);      g2.first = ".com";      g2.second = 2;        cout << "This is pair g" << g1.second << " with "          << "value " << g1.first << "." << endl << endl;        cout << "This is pair g" << g3.second          << " with value " << g3.first          << "This pair was initialized as a copy of "          << "pair g2" << endl << endl;        cout << "This is pair g" << g2.second          << " with value " << g2.first          << "\nThe values of this pair were"          << " changed after initialization."          << endl << endl;        cout << "This is pair g4 with values "          << g4.first << " and " << g4.second          << " made for showing addition. \nThe "          << "sum of the values in this pair is "          << g4.first+g4.second          << "." << endl << endl;        cout << "We can concatenate the values of"          << " the pairs g1, g2 and g3 : "          << g1.first + g3.first + g2.first << endl << endl;        cout << "We can also swap pairs "          << "(but type of pairs should be same) : " << endl;      cout << "Before swapping, " << "g1 has " << g1.first          << " and g2 has " << g2.first << endl;      swap(g1, g2);      cout << "After swapping, "          << "g1 has " << g1.first << " and g2 has " << g2.first;        return 0;  } |

Output:

This is pair g1 with value Geeks.

This is pair g3 with value QuizThis pair was initialized as a copy of pair g2

This is pair g2 with value .com

The values of this pair were changed after initialization.

This is pair g4 with values 5 and 10 made for showing addition.

The sum of the values in this pair is 15.

We can concatenate the values of the pairs g1, g2 and g3 : GeeksQuiz.com

We can also swap pairs (but type of pairs should be same) :

Before swapping, g1 has Geeks and g2 has .com

After swapping, g1 has .com and g2 has Geeks

# 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.

|  |
| --- |
| // 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;  } |

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

40 50

20 60

**Case 3 : Sorting the vector elements on the basis of first element of pairs in descending order.**

For these instances, 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 1st element of pair in  // descending order  #include<bits/stdc++.h>  using namespace std;    // Driver function to sort the vector elements by  // first element of pair in descending order  **bool sortinrev(const pair<int,int> &a,  const pair<int,int> &b)**  **{**  **return (a.first > b.first);**  **}**    int main()  {      // declaring vector of pairs      vector< pair <int,int> > vect;        // initializing 1st and 2nd element of      // pairs with array values      int arr[] = {5, 20, 10, 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 applying sort 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 modified sort() function to sort**  **sort(vect.begin(), vect.end(), sortinrev);**        // Printing the sorted vector(after using sort())      cout << "The vector after applying sort 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 is:

5 30

20 60

10 20

40 50

The vector after applying sort is:

40 50

20 60

10 20

5 30

**Case 4 : Sorting the vector elements on the basis of second element of pairs in descending order.**

These instances can also be handled by modifying “sort()” function and again passing a call to user defined function.

|  |
| --- |
| // C++ program to demonstrate sorting/in vector of  // pair according to 2nd element of pair in  // descending order  #include<bits/stdc++.h>  using namespace std;    // Driver function to sort the vector elements by  // second element of pair in descending order  **bool sortbysecdesc(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;        // Initializing 1st and 2nd element of      // pairs with array values      int arr[] = {5, 20, 10, 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 modified sort() function to sort**  **sort(vect.begin(), vect.end(), sortbysecdesc);**        // Printing the sorted vector(after using sort())      cout << "The vector after applying 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 sort operation is:

5 30

20 60

10 20

40 50

The vector after applying sort operation is:

20 60

40 50

5 30

10 20