**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**

**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');

Another valid syntax to declare pair is:

g2 = {1, 'a'};

* *If not initialized, the first pair values get automatically initialized with value ‘0’ and following pairs not initialized.*

|  |
| --- |
| #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

**operators(=, ==, !=, >=, <=) :** We can use operators with pairs as well.

**using equal(=) :** It assigns new object for a pair object.

#include <iostream>

#include<utility>

using namespace std;

int main()

{ pair<int,char>p1={1,'a'};

pair<int,char>p2;

p2=p1;

cout << p2.first << endl;

cout<<p2.second<<endl;

return 0;

}

**output**

1

a

**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

if(p1==p2)

cout<<"true"<<endl;

**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.  
For pairs like p1=(1,20) and p2=(1,10)   
p2<p1 should give 0 (as it compares 1st element only & they are equal so it’s definitely not less), but that isn’t true. Here the pair compares the second element and if it satisfies then returns 1   
(this is only the case when the first element gets equal while using a relational operator > or < only, otherwise these operators work as mentioned above)

**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.

* CPP

|  |
| --- |
| #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 << "\after 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)

**Vector of Pairs in C++**

A [pair](https://www.geeksforgeeks.org/pair-in-cpp-stl/) is a container which stores two values mapped to each other, and a [vector](https://www.geeksforgeeks.org/vector-in-cpp-stl/) 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.

// Using simple sort() function to sort

    sort(vect.begin(), vect.end());

//use vect.rbegin() and vect.rend() for decreasing order

**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

// 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); // (>) symbol for decreasing order

}

// Using sort() function to sort by 2nd element

// of pair

sort(vect.begin(), vect.end(), sortbysec);