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.

|  |
| --- |
| //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');**

**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**

### **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

### **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

### **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)