



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
```

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
```

2. **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 sec-

ond 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.

```
//C++ 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
```

3. **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 << "\n";
    cout << "Contents of pair2 = " << pair2.first << " " << pair2.second << "\n";
    pair1.swap(pair2);

    cout << "\nAfter swapping:\n ";
    cout << "Contents of pair1 = " << pair1.first << " " << pair1.second << "\n";
    cout << "Contents of pair2 = " << pair2.first << " " << pair2.second << "\n";

    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 val-

ues 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

This article is contributed by **MAZHAR IMAM KHAN**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recommended Posts:

[Set in C++ Standard Template Library \(STL\)](#)

[The C++ Standard Template Library \(STL\)](#)

[Map in C++ Standard Template Library \(STL\)](#)

[Deque in C++ Standard Template Library \(STL\)](#)

[Queue in Standard Template Library \(STL\)](#)

[List in C++ Standard Template Library \(STL\)](#)

[Multiset in C++ Standard Template Library \(STL\)](#)

[Multimap in C++ Standard Template Library \(STL\)](#)

[Sort in C++ Standard Template Library \(STL\)](#)

[Binary Search in C++ Standard Template Library \(STL\)](#)

[Unordered Sets in C++ Standard Template Library](#)

Priority Queue in C++ Standard Template Library (STL)
std::tuple, std::pair | Returning multiple values from a function using Tuple and Pair in C++
is_pointer Template in C++
std is_floating_point Template in C++

Improved By : papun007

Article Tags : C++ cpp-containers-library cpp-pair STL

Practice Tags : STL CPP



48

2.2

☐ To-do ☐ Done

Based on 77 vote(s)

Feedback/ Suggest Improvement Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

[About Us](#)
[Careers](#)
[Privacy Policy](#)
[Contact Us](#)

LEARN

[Algorithms](#)
[Data Structures](#)
[Languages](#)
[CS Subjects](#)
[Video Tutorials](#)

PRACTICE

[Courses](#)
[Company-wise](#)
[Topic-wise](#)
[How to begin?](#)

CONTRIBUTE

[Write an Article](#)
[Write Interview Experience](#)
[Internships](#)
[Videos](#)



@geeksforgeeks, Some rights reserved