Pair in C++ STL

The pair container is a simple container defined in 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;
}

```

Run

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;
}

```

Run

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;
}

```

Run

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 << " " ;
   ```

```cpp
    cout << PAIR3.second << endl ;

    return 0;
}

```

Run

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 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

// 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;
}

```

Run

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.

```cpp

#include <iostream>
```

```cpp
#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;
}

```

Run

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

// 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."
```

Run

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
```

Sample Problem : Sort an array according to another Array

Problem: Given two arrays a[] and b[] of equal size. The task is to sort the array b[] according to the elements of array a[]. That is, elements of the array b[] should be rearranged by following the corresponding elements of array a[] as appeared in sorted order.

Example:

```
Input: a[] = {2, 1, 5, 4, 8, 3, 6, 7};
       b[] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'}
Output: B A F D C G H E
Explanation:
Consider first elements of both arrays: (2, A)
Since the correct location of element 2 in a[] is at position 2.
Therefore, the corresponding element of b[] is also placed at position 2.
Similarly, the rest of the elements are arranged in the following way:
(1, B)
(2, A)
(3, F)
(4, D)
(5, C)
(6, G)
(7, H)
(8, E)
```

Solution: Consider the elements in the both arrays as pairs. That is, in the i-th pair, first element is the element at the i-th index of array a[] and the second element will be the element at the i-th index of array b[].

- Create an array of Pairs of the same size as that of input arrays.
- Now fill the array of pairs as stated above.
- Since we need to arrange the second element of pairs following a sorted order of the first element, we can simply use the sort() function of C++ STL to do this.
- The sort() function will automatically arrange all of the pairs following a sorted order of the first element. We will learn about sort() in C++ in detail later.

Below is the implementation of the above approach:

```

// Program to implement sort an array
// according to another array

#include<bits/stdc++.h>
using namespace std;

// Function to sort the elements of array b[]
```

```cpp
// according to the elements of array a[]
void pairSort(int a[], char b[], int n)
{
    // Create an array of pairs
    pair<int, int> arr_p[n];

    // Fill the array of pairs such that
    // first element of pair is the elements of a[]
    // second element of pair is the corresponding
    // element of array b[]
    for(int i = 0; i < n; i++)
    {
        arr_p[i].first = a[i];
        arr_p[i].second = b[i];
    }

    // Sort the array of pairs
    // By default, the sort function sorts a
    // container by first element
    sort(arr_p, arr_p + n);

    for(int i = 0; i < n; i++)
```

Run

Output:

```
(1, B) (2, A) (3, F) (4, D) (5, C) (6, G) (7, H) (8, E)
```