Pair .. Artful

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

**Member Functions**

1. **make\_pair()** : This template function allows to create a value pair without writing the types explicitly.
2. **swap :**This function swaps the contents of one pair object with the contents of another pair object. The pairs must be of same type.

**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.

pair <string,double> product1; // default constructor

pair <string,double> product2 ("tomatoes",2.30); // value init

pair <string,double> product3 (product2); // copy constructor

product1 = make\_pair(string("lightbulbs"),0.99); // using make\_pair (move)

product2.first = "shoes"; // the type of first is string

product2.second = 39.90; // the type of second is double

cout << "The price of " << product1.first << " is $" << product1.second <<endl;

cout << "The price of " << product2.first << " is $" << product2.second << endl;

cout << "The price of " << product3.first << " is $" << product3.second << endl;

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 << PAIR2.first ; //it prints nothing i.e NULL

cout << PAIR2.second ; //it prints nothing i.e NULL

pair <string, double> PAIR3 ;

PAIR3 = make\_pair ("All the best ",4.56);

cout << PAIR3.first << " " ;

cout << PAIR3.second << endl ;

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;

pair<char, int>pair11 = make\_pair('A', 1);

pair<char, int>pair22 = make\_pair('B', 2);

cout << "Before swapping:\n " ;

cout << "Contents of pair1 = " << pair11.first << " " << pair1.second ;

gap

cout << "Contents of pair2 = " << pair22.first << " " << pair2.second ;

pair1.swap(pair2);

Output

The price of lightbulbs is $0.99

The price of shoes is $39.9

The price of tomatoes is $2.3

00

All the best 4.56

0

1

0

1

0

1

Before swapping:

Contents of pair1 = A 12

Contents of pair2 = B 12