6-关联式容器-map&set

----------比特科技整理----------

map和set底层都是RBTree。RBTree我们在数据结构课程部分已经学习并实现。这里我们主要关注map和set封装的接口及使用。

- map的基本使用

http://www.cplusplus.com/reference/map/map/
http://www.cplusplus.com/reference/map/multimap/

Capacity:

empty	Test whether container is empty (public member function
size	Return container size (public member function)
max_size	Return maximum size (public member function)

Element access:

operator[]	Access element (public member function)
at C++11	Access element (public member function)

Modifiers:

insert	Insert elements (public member function)
erase	Erase elements (public member function)
swap	Swap content (public member function)
clear	Clear content (public member function)

- set

http://www.cplusplus.com/reference/set/set/
http://www.cplusplus.com/reference/set/multiset/

Modifiers:

insert	Insert element (public member function)
erase	Erase elements (public member function)
swap	Swap content (public member function)
clear	Clear content (public member function)
emplace C++11	Construct and insert element (public member function)
emplace_hint C++11	Construct and insert element with hint (public member function)

Observers:

key_comp	Return comparison object (public member function)
value_comp	Return comparison object (public member function)

Operations:

find	Get iterator to element (public member function)
count	Count elements with a specific value (public member function)

- map和set底层的实现

map和set底层都是红黑树，也就是说它增删查改的效率都是O(lgN)

```
template<class K, class V>
```

```
struct Pair
{
    K _first ;
    V _second ;

    Pair(const K& K, const V& v)
        : _first(k )
        , _second(v )
    {}
};

template <class K, class V, class Compare = Less<K >>
class Map
{
public :
    Pair<Iterator , bool> Insert(const Pair< K,V >& pr);
    V& operator [](const K& key );
    Iterator Find (const K& key );
    void Erase (const K& key );
    void Erase (const Iterator& pos );
    Iterator Begin ();
    Iterator End ();
protected :
    RBTree<K , V, Compare> _t ;
};

template <class K, class Compare = Less<K >>
class Set
{
public :
    Pair<Iterator , bool> Insert(const K& key);
    Iterator Find (const K& key );
    void Erase (const K& key );
    void Erase (const Iterator& pos );
    Iterator Begin ();
    Iterator End ();
protected :
    RBTree<K , K, Compare> _t ;
};
```