C++ unordered_map emplace()和emplace_hint()方法

和前面学的 map、set 等容器一样，C++ 11 标准也为 unordered_map 容器新增了 emplace() 和 emplace_hint() 成员方法，本节将对它们的用法做详细的介绍。

我们知道，实现向已有 unordered_map 容器中添加新键值对，可以通过调用 insert() 方法，但其实还有更好的方法，即使用 emplace() 或者 emplace_hint() 方法，它们完成“向容器中添加新键值对”的效率，要比 insert() 方法高。

> 至于为什么 emplace()、emplace_hint() 执行效率会比 insert() 方法高，可阅读《为什么emplace()、emplace_hint()执行效率比insert()高》一文，虽然此文的讲解对象为 map 容器，但就这 3 个方法来说，unordered_map 容器和 map 容器是一样的。

unordered_map emplace()方法

emplace() 方法的用法很简单，其语法格式如下：

```
template <class... Args>
  pair<iterator, bool> emplace ( Args&&... args );
```

其中，参数 args 表示可直接向该方法传递创建新键值对所需要的 2 个元素的值，其中第一个元素将作为键值对的键，另一个作为键值对的值。也就是说，该方法无需我们手动创建键值对，其内部会自行完成此工作。

另外需要注意的是，该方法的返回值为 pair 类型值，其包含一个迭代器和一个 bool 类型值：

- 当 emplace() 成功添加新键值对时，返回的迭代器指向新添加的键值对，bool 值为 True；
- 当 emplace() 添加新键值对失败时，说明容器中本就包含一个键相等的键值对，此时返回的迭代器指向的就是容器中键相同的这个键值对，bool 值为 False。

举个例子：

```
01. #include <iostream>
02. #include <string>
03. #include <unordered_map>
04. using namespace std;
05. int main()
06. {
07.     //创建 umap 容器
08.     unordered_map<string, string> umap;
09.     //定义一个接受 emplace() 方法的 pair 类型变量
10.     pair<unordered_map<string, string>::iterator, bool> ret;
```

```
    //调用 emplace() 方法
    ret = umap.emplace("STL教程", "http://c.biancheng.net/stl/");
    //输出 ret 中包含的 2 个元素的值
    cout << "bool =" << ret.second << endl;
    cout << "iter ->" << ret.first->first << " " << ret.first->second << endl;
    return 0;
}
```

程序执行结果为：

```
bool =1
iter ->STL教程 http://c.biancheng.net/stl/
```

通过执行结果中 bool 变量的值为 1 可以得知，emplace() 方法成功将新键值对添加到了 umap 容器中。

unordered_map emplace_hint()方法

emplace_hint() 方法的语法格式如下：

```
template <class... Args>
    iterator emplace_hint ( const_iterator position, Args&&... args );
```

和 emplace() 方法相同，emplace_hint() 方法内部会自行构造新键值对，因此我们只需向其传递构建该键值对所需的 2 个元素（第一个作为键，另一个作为值）即可。不同之处在于：

- emplace_hint() 方法的返回值仅是一个迭代器，而不再是 pair 类型变量。当该方法将新键值对成功添加到容器中时，返回的迭代器指向新添加的键值对；反之，如果添加失败，该迭代器指向的是容器中和要添加键值对键相同的那个键值对。
- emplace_hint() 方法还需要传递一个迭代器作为第一个参数，该迭代器表明将新键值对添加到容器中的位置。需要注意的是，新键值对添加到容器中的位置，并不是此迭代器说了算，最终仍取决于该键值对的键的值。

> 可以这样理解，emplace_hint() 方法中传入的迭代器，仅是给 unordered_map 容器提供一个建议，并不一定会被容器采纳。

举个例子：

```
#include <iostream>
#include <string>
#include <unordered_map>
using namespace std;
int main()
```

```
{
    //创建 umap 容器
    unordered_map<string, string> umap;
    //定义一个接受 emplace_hint() 方法的迭代器
    unordered_map<string,string>::iterator iter;
    //调用 empalce_hint() 方法
    iter = umap.emplace_hint(umap.begin(),"STL教程", "http://c.biancheng.net/stl/");
    //输出 emplace_hint() 返回迭代器 iter 指向的键值对的内容
    cout << "iter ->" << iter->first << " " << iter->second << endl;
    return 0;
}
```

程序执行结果为：

```
iter ->STL教程 http://c.biancheng.net/stl/
```