c++ unique函数

unique函数　属于STL中比较常用函数，它的功能是元素去重。即”删除”序列中所有相邻的重复元素(只保留一个)。此处的删除，并不是真的删除，而是指重复元素的位置被不重复的元素给占领了(详细情况，下面会讲)。由于它”删除”的是相邻的重复元素，所以在使用unique函数之前，一般都会将目标序列进行排序。

函数原型

unique函数的函数原型如下：

只有两个参数，且参数类型都是迭代器：

```
iterator unique(iterator it_1,iterator it_2);
```

这种类型的unique函数是我们最常用的形式。其中这两个参数表示对容器中[it_1，it_2)范围的元素进行去重(注：区间是前闭后开，即不包含it_2所指的元素),返回值是一个迭代器，它指向的是去重后容器中不重复序列的最后一个元素的下一个元素。

函数用法实例

上面介绍了unique函数的功能和原型，那么，它到底是如何进行去重的呢？即“删除”的具体操作是怎样的呢？

unique函数是完全等价于下面这个函数的：

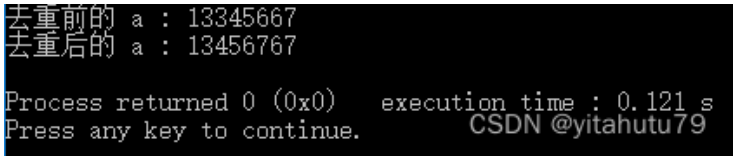
```
iterator My_Unique (iterator first, iterator last)
{
  if (first==last) return last;

  iterator result = first;
  while (++first != last)
  {
    if (!(*result == *first))
      *(++result)=*first;
  }
  return ++result;
}
```

分析这段代码，我们可以知道，unique函数的去重过程实际上就是不停的把后面不重复的元素移到前面来，也可以说是用不重复的元素占领重复元素的位置。有了这段代码我们可以结合实例来更好的理解这个函数了。

实例：

```
#include<iostream>
#include<algorithm>
#include<cassert>
using namespace std;

static bool myfunc(int i, int j)
{
    return (i + 1) == j;
    //return i == j;
}
int main()
{

    vector<int> a = {1,3,3,4,5,6,6,7};
    vector<int>::iterator it_1 = a.begin();
    vector<int>::iterator it_2 = a.end();

    //sort(it_1,it_2);
    cout<<"去重前的 a : ";
    for(int i = 0 ; i < a.size(); i++)
        cout<<a[i];
    cout<<endl;
    //it_h = unique(it_1,it_2);
    //unique(it_1,it_2,myfunc);
    unique(it_1,it_2);
    cout<<"去重后的 a : ";
    for(int i = 0 ; i < a.size(); i++)
        cout<<a[i];
    cout<<endl;
}

```

运行结果如下：



对于上面的结果，我们可以看到，容器中不重复的元素都移到了前面，至于后面的元素，实际上并没有改变(这个过程只需结合My_Unique函数来分析即可)。

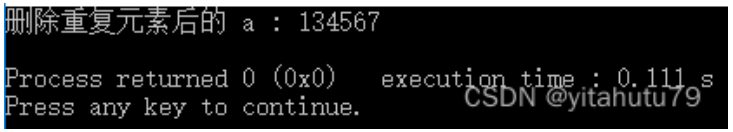
> 注：
> 1.有很多文章说的是，unique去重的过程是将重复的元素移到容器的后面去，实际上这种说法并不正确，应该是把不重复的元素移到前面来。
> 2.一定不要忘记的是，unique函数在使用前需要对容器中的元素进行排序(当然不是必须的，但我们绝大数情况下需要这么做)，由于本例中的元素已经是排好序的，所以此处我没排序，但实际使用中不要忘记。

unique函数通常和erase函数一起使用，来达到删除重复元素的目的。(注：此处的删除是真正的删除，即从容器中去除重复的元素，容器的长度也发生了变换；而单纯的使用unique函数的话，容器的长度并没有发生变化，只是元素的位置发生了变化)关于erase函数的用法。下面是一个具体的实例：

```
#include<iostream>
#include<algorithm>
#include<cassert>
using namespace std;

int main()
{

    vector<int> a ={1,3,3,4,5,6,6,7};
    vector<int>::iterator it_1 = a.begin();
    vector<int>::iterator it_2 = a.end();
    vector<int>::iterator new_end;

    new_end = unique(it_1,it_2); //注意unique的返回值
    a.erase(new_end,it_2);
    cout<<"删除重复元素后的 a : ";
    for(int i = 0 ; i < a.size(); i++)
        cout<<a[i];
```

```
    cout<<endl;
}
```

运行结果如下：



可以看到，相比之前的结果，a的长度确实发生了改变，真正的删除了a中的重复元素。