

## 转 STL的Vector介绍

2013年09月24日 12:30:59 阅读数：1982

Vector在使用的时候比数组要方便得多，自从学会使用后，我基本上使用Vector代替了数组，编程效率提高了很多。在此进行简单的介绍。

vector 是同一种类型的对象的集合,每个对象都有一个对应的整数索引值。

和 string 对象一样,标准库将负责管理与存储元素相关的内存。我们把 vector称为容器,是因为它可以包含其他对象。一个容器中的所有对象都必须是同一种类型的。vector 是一个类模板(class template)。使用模板可以编写一个类定义或函数定义,而用于多个不同的数据类型。因此,我们可以定义保存 string 对象的 vector,或保存 int 值的 vector,又或是保存自定义的类类型对象(如Sales\_items 对象)的 vector。vector 不是一种数据类型,而只是一个类模板,用来定义任意多种数据类型。vector 类型的每一种都指定了其保存元素的类型。

vector之所以被认为是一个容器,是因为它能够像容器一样存放各种类型的对象,简单地说, vector是一个能够存放任意类型的动态数组,能够增加和压缩数据。

为了可以使用vector, 必须在你的头文件中包含下面的代码：

```
1. #include <vector>
```

vector属于std命名域的, 因此需要通过命名限定, 如下完成你的代码：

```
1. using std::vector;
2. vector<int> vInts;
```

或者连在一起, 使用全名：

std::vector<int> vInts;

建议在代码量不大, 并且使用的命名空间不多的情况下, 使用全局的命名域方式：using namespace std;

函数

表述

c.assign(beg,end) c.assign(n,elem)

将 (beg; end) 区间中的数据赋值给c。将n个elem的拷贝赋值给c。

传回索引idx所指的数据, 如果idx越界, 抛出out\_of\_range。

c.back()

传回最后一个数据, 不检查这个数据是否存在。

c.begin()

传回迭代器中的第一个数据地址。

c.capacity()

返回容器中数据个数。

```
1. c.clear()
```

移除容器中所有数据。

```
1. c.empty()
```

判断容器是否为空。

c.end() //指向迭代器中末端元素的下一个, 指向一个不存在元素。

c.erase(pos)// 删除pos位置的数据, 传回下一个数据的位置。

c.erase(beg,end)

删除[beg,end) 区间的数据, 传回下一个数据的位置。

c.front()

传回第一个数据。

get\_allocator

使用构造函数返回一个拷贝。

c.insert(pos,elem)//在pos位置插入一个elem拷贝, 传回新数据位置

c.insert(pos,n,elem)//在pos位置插入n个elem数据, 无返回值

c.insert(pos,beg,end)//在pos位置插入在[beg,end) 区间的数据。无返回值

c.max\_size()

返回容器中最大数据的数量。

```
1. c.pop_back()
```

删除最后一个数据。

```
1. c.push_back(elem)
```

在尾部加入一个数据。

c.rbegin()

传回一个逆向队列的第一个数据。

c.rend()

传回一个逆向队列的最后一个数据的下一个位置。

`c.resize(num)`

重新指定队列的长度。

`c.reserve()`

保留适当的容量。

```
[cpp]
1. c.size()
```

返回容器中实际数据的个数。

`c1.swap(c2)`//将c1和c2元素互换

`swap(c1,c2)`//同上操作。

`vector<Elem>` //创建一个空的vector

`vector<Elem> c1(c2)`//复制一个vector

`vector<Elem> c(n)`//创建一个vector，含有n个数据，数据均已缺省构造产生

`vector<Elem> c(n,elem)`//创建一个含有n个elem拷贝的vector

`vector<Elem> c(beg,end)`//创建一个以 (beg;end) 为区间的vector

`c.~ vector<Elem>()`//销毁所有数据，释放内存

`operator[]`

返回容器中指定位置的一个引用。

创建一个vector

vector容器提供了多种创建方法，下面介绍几种常用的。

创建一个Widget类型的空的vector对象：

`vector<Widget> vWidgets;`

创建一个包含500个Widget类型数据的vector：

`vector<Widget> vWidgets(500);`

创建一个包含500个Widget类型数据的vector，并且都初始化为0：

`vector<Widget> vWidgets(500,Widget(0));`

创建一个Widget的拷贝：

`vector<Widget> vWidgetsFromAnother(vWidgets);`

向vector添加一个数据

vector添加数据的缺省方法是`push_back ()`。`push_back ()`函数表示将数据添加到vector的尾部，并按需要来分配内存。例如：向`vector<Widget>`；中添加10个数据，需要如下编写代码：

```
for(int i= 0;i<10; i++) {
vWidgets.push_back(Widget(i));
}
```

获取vector中指定位置的数据

vector里面的数据是动态分配的，使用`push_back()`的一系列分配空间常常决定于文件或一些数据源。如果想知道vector是否为空，可以使用`empty ()`，空返回true，否则返回false。获取vector的大小，可以使用`size ()`。例如，如果想获取一个vector v的大小，但不知道它是否为空，或者已经包含了数据，如果为空时想设置为 -1，你可以使用下面的代码实现：

```
int nSize = v.empty() ? -1 : static_cast<int>(v.size());[3]
```

访问vector中的数据

使用两种方法来访问vector。

```
[cpp]
1. 1、 vector::at()
2. 2、 vector::operator[]
```

`operator[]`主要是为了与C语言进行兼容。它可以像C语言数组一样操作。但`at ()`是我们的首选，因为`at ()`进行了边界检查，如果访问超过了vector的范围，将抛出一个例外。由于`operator[]`容易造成一些错误，所以我们很少用它。

删除vector中的数据

vector能够非常容易地添加数据，也能很方便地取出数据，同样vector提供了`erase ()`，`pop_back ()`，`clear ()`来删除数据，当删除数据时，应该知道要删除尾部的数据，或者是删除所有数据，还是个别的数据。

文章标签：[stl](#) [vector](#)

个人分类：[纯编程](#)

此PDF由spygg生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com