每个容器都有很多操作，以下操作为所有容器共有的**：**

**一．构造，拷贝和析构**

|  |  |
| --- | --- |
| **ContType <Elem> c** | 创建一个空容器，其中没有任何元素 |
| **ContType <Elem> c1(c2)** | 创建一个同种类形容器的一个copy |
| **ContType <Elem> c(beg,end)** | 创建一个容器，以区间[beg,end)做为元素初值 |
| **c.~ContType <Elem>()** | 销毁所有元素，并释放内存 |

**二．与大小相关的操作**

|  |  |
| --- | --- |
| **c.size()** | 返回当前的元素数量 |
| **c.empty()** | 返回 bool 型，判断容量是否为0。 等同与 size()==0, 但可能更快 |
| **c.max\_size()** | 返回可容纳元素的最大数量，很大一个数 |

例子：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7302597)

1. #include <iostream>
2. #include <vector>
3. using namespace std;
4. **int** main(**int** argc, **char**\* argv[])
5. {
6. vector<**int**> col(10);
7. cout << col.size() << endl; //10
8. if (col.empty()) //bool 型
9. {
10. cout << "empty"<<endl;
11. }
12. cout<<col.max\_size()<<endl; //很大一个数
13. return 0;
14. }

#include <iostream>

#include <vector>

using namespace std;

int main(int argc, char\* argv[])

{

vector<int> col(10);

cout << col.size() << endl; //10

if (col.empty()) //bool 型

{

cout << "empty"<<endl;

}

cout<<col.max\_size()<<endl; //很大一个数

return 0;

}

**三．比较操作**

|  |  |
| --- | --- |
| **c1 == c2** | 判断 c1 是否等于 c2 |
| **c1 != c2** | 判断 c1 是否不等于 c2 |
| **c1 < c2** | 判断 c1 是否小于 c2 |
| **c1 > c2** | 判断 c1 是否大于 c2 |
| **c1 <= c2** | 判断 c1 是否小于等于 c2 |
| **c1 >= c2** | 判断 c1 是否大于等于 c2 |

**四．赋值操作**

|  |  |
| --- | --- |
| **c1 = c2** | 将c2的全部元素赋值给c1，c1与c2必须是同种类型 |

**五．交换操作**

|  |  |
| --- | --- |
| **c1.swap(c2)** | 将c1和c2元素互换 |
| **swap(c1,c2)** | 同上，是个全局函数 |

**六．迭代器相关**

|  |  |
| --- | --- |
| **c.begin()** | 返回一个随机存取迭代器，指向第一个元素 |
| **c.end()** | 返回一个随机存取迭代器，指向最后元素的**下一个**位置 |
| **c.rbegin()** | 返回一个逆向迭代器，指向逆向迭代的第一元素，也就是最后一个元素 |
| **c.rend()** | 返回一个逆向迭代器，指向逆向迭代的最后元素的下一个位置，也就是第一个元素的**前一个**位置 |

注意：\*c.end与\*c.rend都是没有定义的。

**七．插入与删除**

|  |  |
| --- | --- |
| **c.insert(pos,elem)** | 在pos位置**前**插入一个elem副本，并返回新元素位置 |
| **c.erase(beg,end)** | 移除 [beg,end) 区间内的所有元素，返回下一元素的位置 |
| **c.clear()** | 移除 所有元素，将容器清空 |