C++ STL vector容器详解

vector 容器是 STL 中最常用的容器之一，它和 array 容器非常类似，都可以看做是对 C++ 普通数组的“升级版”。不同之处在于，array 实现的是静态数组（容量固定的数组），而 vector 实现的是一个动态数组，即可以进行元素的插入和删除，在此过程中，vector 会动态调整所占用的内存空间，整个过程无需人工干预。

vector 常被称为向量容器，因为该容器擅长在尾部插入或删除元素，在常量时间内就可以完成，时间复杂度为 O(1)；而对于在容器头部或者中部插入或删除元素，则花费时间要长一些（移动元素需要耗费时间），时间复杂度为线性阶 O(n)。

> 有关复杂度，可阅读《大O表示法》一节详细了解。

vector 容器以类模板 vector<T>（T 表示存储元素的类型）的形式定义在 <vector> 头文件中，并位于 std 命名空间中。因此，在创建该容器之前，代码中需包含如下内容：

```
01.  #include <vector>
02.  using namespace std;
```

> 注意，std 命名空间也可以在使用 vector 容器时额外注明，两种方式都可以。

创建vector容器的几种方式

创建 vector 容器的方式有很多，大致可分为以下几种。

1) 如下代码展示了如何创建存储 double 类型元素的一个 vector 容器：

```
01.  std::vector<double> values;
```

> 如果程序中已经默认指定了 std 命令空间，这里可以省略 std::。

注意，这是一个空的 vector 容器，因为容器中没有元素，所以没有为其分配空间。当添加第一个元素（比如使用 push_back() 函数）时，vector 会自动分配内存。

在创建好空容器的基础上，还可以像下面这样通过调用 reserve() 成员函数来增加容器的容量：

```
01.  values.reserve(20);
```

这样就设置了容器的内存分配，即至少可以容纳 20 个元素。注意，如果 vector 的容量在执行此语句之前，已经大于或等于 20 个元素，那么这条语句什么也不做；另外，调用 reserve() 不会影响已存储的元素，也不会生成任何元素，即 values 容器内此时仍然没有任何元素。

> 还需注意的是，如果调用 reserve() 来增加容器容量，之前创建好的任何迭代器（例如开始迭代器和结束迭代器）都可能会失效，这是因为，为了增加容器的容量，vector<T> 容器的元素可能已经被复制或移到了新的内存地址。所以后续再使用这些迭代器时，最好重新生成一下。

2) 除了创建空 vector 容器外，还可以在创建的同时指定初始值以及元素个数，比如：

```
01.  std::vector<int> primes {2, 3, 5, 7, 11, 13, 17, 19};
```

这样就创建了一个含有 8 个素数的 vector 容器。

3) 在创建 vector 容器时，也可以指定元素个数：

```
01.  std::vector<double> values(20);
```

如此，values 容器开始时就有 20 个元素，它们的默认初始值都为 0。

> 注意，圆括号 () 和大括号 {} 是有区别的，前者（例如 (20) ）表示元素的个数，而后者（例如 {20} ） 则表示 vector 容器中只有一个元素 20。

如果不想用 0 作为默认值，也可以指定一个其它值，例如：

```
01.  std::vector<double> values(20, 1.0);
```

第二个参数指定了所有元素的初始值，因此这 20 个元素的值都是 1.0。

值得一提的是，圆括号 () 中的 2 个参数，既可以是常量，也可以用变量来表示，例如：

```
01.  int num=20;
02.  double value =1.0;
03.  std::vector<double> values(num, value);
```

4) 通过存储元素类型相同的其它 vector 容器，也可以创建新的 vector 容器，例如：

```
01.  std::vector<char>value1(5, 'c');
02.  std::vector<char>value2(value1);
```

由此，value2 容器中也具有 5 个字符 'c'。在此基础上，如果不想复制其它容器中所有的元素，可以用一对指针或者迭代器来指定初始值的范围，例如：

```
int array[]={1,2,3};
std::vector<int>values(array, array+2);//values 将保存{1,2}
std::vector<int>value1{1,2,3,4,5};
std::vector<int>value2(std::begin(value1),std::begin(value1)+3);//value2保存{1,2,3}
```

由此，value2 容器中就包含了 {1,2,3} 这 3 个元素。

vector容器包含的成员函数

相比 array 容器，vector 提供了更多了成员函数供我们使用，它们各自的功能如表 1 所示。

表 1 vector 容器的成员函数

函数成员	函数功能
begin()	返回指向容器中第一个元素的迭代器。
end()	返回指向容器最后一个元素所在位置后一个位置的迭代器，通常和 begin() 结合使用。
rbegin()	返回指向最后一个元素的迭代器。
rend()	返回指向第一个元素所在位置前一个位置的迭代器。
cbegin()	和 begin() 功能相同，只不过在其基础上，增加了 const 属性，不能用于修改元素。
cend()	和 end() 功能相同，只不过在其基础上，增加了 const 属性，不能用于修改元素。
crbegin()	和 rbegin() 功能相同，只不过在其基础上，增加了 const 属性，不能用于修改元素。
crend()	和 rend() 功能相同，只不过在其基础上，增加了 const 属性，不能用于修改元素。
size()	返回实际元素个数。
max_size()	返回元素个数的最大值。这通常是一个很大的值，一般是 2^{32}-1，所以我们很少会用到这个函数。
resize()	改变实际元素的个数。
capacity()	返回当前容量。

empty()	判断容器中是否有元素，若无元素，则返回 true；反之，返回 false。
reserve()	增加容器的容量。
shrink _to_fit()	将内存减少到等于当前元素实际所使用的大小。
operator[ ]	重载了 [ ] 运算符，可以向访问数组中元素那样，通过下标即可访问甚至修改 vector 容器中的元素。
at()	使用经过边界检查的索引访问元素。
front()	返回第一个元素的引用。
back()	返回最后一个元素的引用。
data()	返回指向容器中第一个元素的指针。
assign()	用新元素替换原有内容。
push_back()	在序列的尾部添加一个元素。
pop_back()	移出序列尾部的元素。
insert()	在指定的位置插入一个或多个元素。
erase()	移出一个元素或一段元素。
clear()	移出所有的元素，容器大小变为 0。
swap()	交换两个容器的所有元素。
emplace()	在指定的位置直接生成一个元素。
emplace_back()	在序列尾部生成一个元素。

除此之外，C++ 11 标准库还新增加了 begin() 和 end() 这 2 个函数，和 vector 容器包含的 begin() 和 end() 成员函数不同，标准库提供的这 2 个函数的操作对象，既可以是容器，还可以是普通数组。当操作对象是容器时，它和容器包含的 begin() 和 end() 成员函数的功能完全相同；如果操作对象是普通数组，则 begin() 函数返回的是指向数组第一个元素的指针，同样 end() 返回指向数组中最后一个元素之后一个位置的指针（注意不是最后一个元素）。

vector 容器还有一个 std::swap(x , y) 非成员函数（其中 x 和 y 是存储相同类型元素的 vector 容器），它和 swap() 成员函数的功能完全相同，仅使用语法上有差异。

如下代码演示了表 1 中部分成员函数的用法：

```
01.  #include <iostream>
02.  #include <vector>
```

```
using namespace std;
int main()
{
    //初始化一个空vector容量
    vector<char>value;
    //向value容器中的尾部依次添加 S、T、L 字符
    value.push_back('S');
    value.push_back('T');
    value.push_back('L');
    //调用 size() 成员函数容器中的元素个数
    printf("元素个数为：%d\n", value.size());
    //使用迭代器遍历容器
    for (auto i = value.begin(); i < value.end(); i++) {
        cout << *i << " ";
    }
    cout << endl;
    //向容器开头插入字符
    value.insert(value.begin(), 'C');
    cout << "首个元素为：" << value.at(0) << endl;
    return 0;
}
```

输出结果为：

```
元素个数为：3
S T L
首个元素为：C
```

> 表 1 中这些成员函数的具体用法，后续学习用到时会具体讲解，感兴趣的读者，也可以通过查阅 STL手册做详细了解。