## STL各种容器的操作：

#### vector的各种函数

##### 构造函数

1. vector()：创建一个空的vector
2. vector(const std::allocator<T>& al)：使用指定的分配器来分配内存。**allocator就是一个内存分配器，vector已经指定了默认的分配器了，不需要我们去主动调用，以后设计allocator直接忽略就可以了，其实这个构造函数只不过是用指定的分配器去创建一个空的vector罢了。**
3. vector(std::vector<T>&& right, const std::allocator<T>& al)：就是移动构造函数，第二个参数表示我们指定分配器。
4. vector(const std::vector<T>& vec, const std::alloctor<T>& al)：就是复制构造函数，分配器可以自己指定，当然，一般来说，vector默认的分配器就够用了。
5. vector(std::initializer\_list<T>& initList, const std::allocator<T>& al)：就是使用initializer\_list来初 始化容器，第二个参数表示我们可以指定分配器。
6. vector(iter first, iter last, const std::allocator<T>& al)：就是容器初始有迭代器[first, last)的内容（**这里使用deque，list的迭代器也可以**），第三个参数还是表示我们可以指定分配器。
7. vector(const size\_t count, const std::alloctor<T>& al)：创建一个vector，元素个数为count。元素均为默认值，如果是普通类型，则赋值为0。如果是类类型，则均使用默认构造函数进行初始化。
8. vector(const size\_t count，const T& t)：创建一个vector，元素个数为count，且值均为t。

##### 增加函数

1. void push\_back(const T& value)：向容器尾部增加一个元素value。
2. void push\_back(T&& value)：向容器尾部增加一个元素value，这不过这次以右值引用的形式添加。
3. std::vector<T>::iterator insert(std::vector<T>::const\_iterator& where, std::initializer\_list<T> initList)：在where迭代器指定的地方添加initList，**返回值为指向新添加的第一个元素的迭代器，insert函数虽然有很多重载，但返回值是完全类似的，所以接下来insert函数的返回值就不介绍了。**
4. std::vector<T>::iterator insert(std::vector<T>::const\_iterator& where, iter first, iter last)：将迭代器[first, last)添加到迭代器where指定的位置。
5. std::vector<T>::iterator insert(std::vector<T>::const\_iterator& where, size\_t count, const int& value)：在where处插入count个value。
6. std::vector<T>::iterator insert(std::vector<T>::const\_iterator& where, const T& value)：在where处插入value。
7. std::vector<T>::iterator insert(std::vector<T>::const\_iterator& where, T&& value)：在where处插入value，只不过这次以右值引用的形式插入了。

##### 删除函数

1. std::vector<T>::iterator erase(std::vector<T>::const\_iterator where)：删除容器迭代器指向的元素。**返回指向被删除元素后面的那个元素的迭代器。**
2. iterator erase(iterator first, iterator last)：删除容器中[first, last)中的元素。**返回指向被删除元素后面的那个元素的迭代器。**
3. void pvoid op\_back()：删除容器中最后一个元素。
4. clear()：删除容器中所有元素。

##### 遍历函数

1. T& at(const size\_t pos)：返回pos位置元素的引用。
2. const T& at(const size\_t pos) const：at函数的常量版本。
3. T& front()：返回首元素的引用。
4. const T& front() const：front函数的常量版本。
5. T& back()：返回尾元素的引用。
6. const T& back() const：back函数的常量版本。
7. std::vector<T>::iterator begin()：返回指向容器第一个元素的迭代器。
8. std::vector<T>::const\_iterator begin() const：begin函数的常量版本。
9. std::vector<T>::const\_iterator cbegin() const：可以主动调用的begin函数的常量版本。
10. std::vector<T>::iterator end()：返回指向容器最后一个元素的下一个元数的迭代器。

end()函数也有两个常量版本，和begin类似，就不写了。

1. std::vector<T>::reverse\_iterator rbegin()：反向迭代器，指向最后一个元素。

同样有两个常量版本。

1. reverse\_iterator rend()：反向迭代器，指向第一个元素之前的元素。

同样有两个常量版本。

##### 判断函数

1. bool empty() const：判断容器是否为空，若未空，则返回true，否则返回false。

##### 大小函数

1. size\_t size() const：返回当前容器中元素的个数。
2. size\_t capacity() const：返回当前容器不扩张所能容纳的最大元素数量。
3. size\_t max\_size() const：返回当前机器可以存储的元素数量最大值。

##### 其它函数

1. void swap(std::vector<T> & vec)：交换两个同类型容器的的数据。
2. void assign(int n, const T& x)：将容器设置为n个x。
3. void assign(const\_iterator first, const iterator last)：将当前容器的元素设置为[first, last)。

first,last都是迭代器，可以不是vector类型的迭代器，deque，list类型也可以。

1. void assign(const std::initialize\_list<T> initList)：将容器元素设置为initialize\_list的元素。
2. void res(size\_t newSize)：将容器的容量设置为newSIze。

#### deque的各种函数：

deque的各种函数与vector类似，我就不再重复一遍了。

这里只介绍vector不同的地方：

1. deque支持在容器前面插入删除，操作。也就是支持以下的三个函数
   1. void push\_front(const T& value）；
   2. void push\_front(T&& value)；
   3. void pop\_front()；

#### list的各种函数

list和deque类似，只讲一下和deque不同的部分。list都用支持在前面，后面增加，删除。

list和deque在函数上的唯一区别就是不支持随机缩影，也**就是不支持at函数**。

#### string的各种函数

string虽然也是顺序容器，但因为本质是对字符串的封装，所以和其它容器在用法上有较大区别。

1. 获取封装字符串的函数。
   1. const char\* c\_str() const：返回string对象内部的函数的指针**。注意，c\_str()函数返回的直接就是string对象内部的指针，也就是说string对象指向的对象发生了改变，返回的对象也会发生改变的。**

**代码演示一下：**

* 1. const char\* data()const：返回string对象内部的函数指针。和c\_str()函数的区别就是返回的字符串后面没有’\0’。
  2. size\_t copy(char\* const ptr, size\_t count, const size\_t off) const：

讲string对象的一部分复制到ptr数组中。

ptr表示复制到哪个数组。

count表示复制string对象的几个字符

off表示从string的哪个字符开始复制。

1. 字符串比较函数。

compare函数：这个函数重载比较多，用的时候在vs中查看一下就可以了。可以用string对象的任意部分与另一个字符串进行比较，

其它函数就和vector类似了，同样支持随机选取，支持容器末尾插入。

#### forward\_list:

和list差不多，只不过是没有size()函数，没有push\_back和pop\_back函数。

#### 关联容器的函数和顺序容器有差别的也就那么几个

在讲map，set时基本已经讲过了，这里就不单独讲了。