C++ Vectors

Vectors 包含着一系列连续存储的元素,其行为和数组类似。访问 Vector 中的任意元素或从末尾添加元素都可以在**常量级时间复杂度**内完成，而查找特定值的元素所处的位置或是在 Vector 中插入元素则是**线性时间复杂度**。

创建 Vectors 对象的三种形式：
（1）： vectors <int>v;　　/
（2）:vectors <int>v(10);　　//大小空间为十
（3）:vectors <double>v(10,8.6);　　//　每个元素的初值为 8.6

可用**下标访问元素**：cout<<v[0]<<endl;

Constructors	构造函数
Operators	对 vector 进行赋值或比较
assign()	对 Vector 中的元素赋值
at()	返回指定位置的元素
back()	返回最末一个元素
begin()	返回第一个元素的迭代器
capacity()	返回 vector 所能容纳的元素数量(在不重新分配内存的情况下）
clear()	清空所有元素
empty()	判断 Vector 是否为空（返回 true 时为空）
end()	返回最末元素的迭代器(译注:实指向最末元素的下一个位置)
erase()	删除指定元素
front()	返回第一个元素
get_allocator()	返回 vector 的内存分配器
insert()	插入元素到 Vector 中
max_size()	返回 Vector 所能容纳元素的最大数量（上限）
pop_back()	移除最后一个元素
push_back()	在 Vector 最后添加一个元素
rbegin()	返回 Vector 尾部的逆迭代器
rend()	返回 Vector 起始的逆迭代器
reserve()	设置 Vector 最小的元素容纳数量
resize()	改变 Vector 元素数量的大小
size()	返回 Vector 元素数量的大小
swap()	交换两个 Vector

构造函数

```
vector();
vector( size_type num, const TYPE &val );
vector( const vector &from );
vector( input_iterator start, input_iterator end );
```

C++ Vectors 可以使用以下任意一种参数方式构造:

- 无参数 - 构造一个空的 vector,
- 数量(num)和值(val) - 构造一个初始放入 num 个值为 val 的元素的 Vector
- vector(from) - 构造一个与 vector from 相同的 vector
- 迭代器(start)和迭代器(end) - 构造一个初始值为[start,end)区间元素的 Vector(注:半开区间).

运算符

```
v1 == v2
v1 != v2
v1 <= v2
v1 >= v2
v1 < v2
v1 > v2
v[]
```

C++ Vectors 能够使用标准运算符: ==, !=, <=, >=, <, 和 >. 要访问 vector 中的某特定位置的元素可以使用 [] 操作符.

两个 vectors 被认为是相等的, 如果:

1. 它们具有相同的容量
2. 所有相同位置的元素相等.

vectors 之间大小的比较是按照词典规则.

assign 函数

```
void assign( input_iterator start, input_iterator end );
void assign( size_type num, const TYPE &val );
```

assign() 函数要么将区间[start, end)的元素赋到当前 vector, 或者赋 num 个值为 val 的元素到 vector 中. 这个函数将会清除掉为 vector 赋值以前的内容.

at 函数

```
TYPE at( size_type loc );
```

at() 函数 返回当前 Vector 指定位置 *loc 的元素的引用*. at() 函数 比 [] 运算符更加安全, 因为它不会让你去访问到 Vector 内越界的元素.

back 函数

```
TYPE back();
```

back() 函数返回当前 vector 最末一个元素的引用.

begin 函数

```
iterator begin();
```

begin()函数返回一个指向当前 vector 起始元素的迭代器.

capacity 函数

```
size_type capacity();
```

capacity() 函数 返回当前 vector 在重新进行内存分配以前所能容纳的元素数量.

clear 函数

```
void clear();
```

clear()函数删除当前 vector 中的所有元素.

empty 函数

```
bool empty();
```

如果当前 vector 没有容纳任何元素,则 empty()函数返回 true,否则返回 false.

end 函数

```
iterator end();
```

end() 函数返回一个指向当前 vector 末尾元素的**下一位置**的迭代器. 注意, 如果你要访问末尾元素, 需要先将此迭代器自减 1.

erase 函数

```
iterator erase( iterator loc );
iterator erase( iterator start, iterator end );
```

erase 函数要么删作指定位置 loc 的元素, 要么删除区间[start, end)的所有元素. 返回值是指向删除的最后一个元素的下一位置的迭代器.

front 函数

```
TYPE front();
```

front()函数返回当前 vector 起始元素的引用

get_allocator 函数

```
allocator_type get_allocator();
```

get_allocator() 函数返回当前 vector 的内存分配器.

insert 函数

```
iterator insert( iterator loc, const TYPE &val );
void insert( iterator loc, size_type num, const TYPE &val );
void insert( iterator loc, input_iterator start, input_iterator end );
```

insert() 函数有以下三种用法:

- 在指定位置 loc 前插入值为 val 的元素,返回指向这个元素的迭代器,
- 在指定位置 loc 前插入 num 个值为 val 的元素
- 在指定位置 loc 前插入区间[start, end)的所有元素 .

max_size 函数

```
size_type max_size();
```

max_size() 函数返回当前 vector 所能容纳元素数量的最大值(译注:包括可重新分配内存).

pop_back

```
void pop_back();
```

pop_back()函数删除当前 vector 最末的一个元素

push_back 函数

```
void push_back( const TYPE &val );
```

push_back()添加值为 val 的元素到当前 vector 末尾

rbegin 函数

```
reverse_iterator rbegin();
```

rbegin 函数返回指向当前 vector 末尾的逆迭代器.(译注:实际指向末尾的下一位置,而其内容为末尾元素的值,详见逆代器相关内容)

rend 函数

```
reverse_iterator rend();
```

rend()函数返回指向当前 vector 起始位置的逆迭代器.

reserve 函数

```
void reserve( size_type size );
```

reserve()函数为当前 vector 预留至少共容纳 size 个元素的空间.(译注:实际空间可能大于 size)

resize 函数

```
void resize( size_type size, TYPE val );
```

resize() 函数改变当前 vector 的大小为 size,且对新创建的元素赋值 val

size 函数

```
size_type size();
```

size() 函数返回当前 vector 所容纳元素的数目

swap 函数

```
void swap( vector &from );
```

swap()函数交换当前 vector 与 vector from 的元素