动态扩展：并不是在原空间之后续接新空间，而是找更大的内存空间，然后将原数据拷贝新空间，释放原空间

v.rend()

v.begin()//常用

v.rbegin()

v.end()//常用

vector容器的迭代器是支持随机访问的迭代器（随机访问的迭代器是最强大的）

vector构造函数

功能描述：

创建vector容器

函数原型：

vector<T> v; //采用模板实现类实现，默认构造函数

vector(v.begin(), v.end()); //将v[begin(), end())区间中的元素拷贝给本身。

vector(n, elem); //构造函数将n个elem拷贝给本身

vector(const vector& vec); //拷贝构造函数

总结：vector的多种构造方式没有可比性，灵活使用即可。

vector赋值操作

功能描述：

给vector容器进行赋值

函数原型：  
 vector& operator=(const vector &vec); //重载等号操作符

assign(beg, end); //将[beg, end)区间中的数据拷贝赋值给本身。

assign(n, elem); //将n个elem拷贝赋值给本身。

vector容量和大小

功能描述：

对 vector 容器的容量和大小操作

函数原型

empty(); // 判断容器是否为空

capacity(); // 容器的容量

size(); // 返回容器中元素的个数

resize(int num); // 重新指定容器的长度为 num，若容器变长，则以默认值填充新位置。如果容器变短，则末尾超出容器长度的元素被删除。

resize(int num, elem); // 重新指定容器的长度为 num，若容器变长，则以 elem 值填充新位置。如果容器变短，则末尾超出容器长度的元素被删除

vector 插入和删除

功能描述：

对 vector 容器进行插入、删除操作

函数原型：

push\_back(ele); // 尾部插入元素 ele

pop\_back(); // 删除最后一个元素

insert(const\_iterator pos, ele); // 迭代器指向位置 pos 插入元素 ele

insert(const\_iterator pos, int count,ele); // 迭代器指向位置 pos 插入 count 个元素 ele

erase(const\_iterator pos); // 删除迭代器指向的元素

erase(const\_iterator start, const\_iterator end); // 删除迭代器从 start 到 end 之间的元素

clear(); // 删除容器中所有元素

vector 数据存取

功能描述：

对 vector 中的数据的存取操作

函数原型：

at(int idx); // 返回索引 idx 所指的数据

operator[]; // 返回索引 idx 所指的数据

front(); // 返回容器中第一个数据元素

back(); // 返回容器中最后一个数据元素

vector 互换容器

功能描述：

实现两个容器内的元素交换

函数原型：

swap(vec); //将vec与本身的元素互换

总结：swap可以使两个容器互换，可以达到实用的收缩内存的效果

vector预留空间

功能描述：

减少vector在动态扩展容量时的扩展次数

函数原型：

reserve(int len);//容器预留len个元素长度，预留位置不初始化，元素不可访问