**vector类型**

vector是同一种类型的对象的集合，每一个对象都有一个对应的整数索引值。

vector是一种容器，它可以包含其他对象。一个容器中的所有对象都必须是同一种类型的。

头文件：#include<vector>

vector还是一个类模板，使用模板可以编写一个类定义或者函数定义，而用于多个不同的数据类型。

vector<int> ivec;

vector<Sales\_item> Sales\_vec;

vector对象的定义和初始化

定义了好几个构造函数->定义和初始化vector对象。

创建确定个数的元素

创建非空的vector对象，须初始化元素的值。

vector<int> ivec1;

vector<int> ivec2(ivec1);

用元素个数和元素值对vector对象进行初始化。

vector<int> ivec4(10, -1); // 10 elements, each initialized to -1

vector<string> svec(10, "hi!"); // 10 strings, each initialized to "hi!"

vector对象动态增长

先初始化一个空vector对象，然后再动态地增加元素。

vector.empty()：如果v为空，则返回true，否则返回false

v.size()：返回v中元素的个数，返回相应vector类定义的size\_type的值，vector<int>::size\_type

v.push\_back(t)：在v的末尾增加一个值为t的元素

v[n]：返回v中位置为n的元素

v1 = v2：把v1的元素替换为v2中的元素的副本

v1 == v2：如果v1与v2相等，则返回true

!=, <, <=, >, >=

push\_back()接受一个元素的值，将它作为新的元素添加到vector对象的后面

string word;

vector<string> text;

while(cin>>word) text.push\_back(word);

创建动态数组

与数组变量不同，动态分配的数组将一直存在，直到程序显式释放它为止。

每一个程序在执行时，都占用一块可用的内存空间，用于存放动态分配的对象，此内存空间称为自由存储区(free store)或堆(heap)。

C：malloc和free

C++：new和delete

fill

将一个区间的元素都赋予val值。函数参数：fill(first, last, val); //first为容器的首迭代器，last为容器的末迭代器，val为将要替换的值。

fill\_n

给你一个起始点，然后再给你一个数值count和val，把从起始点开始依次赋予count个元素val的值。

注意： 不能在没有元素的空容器上调用fill\_n函数。

迭代器(iterator)

检查容器内元素并遍历元素的数据类型

vector<int>::iterator iter;

每个容器都定义了一个名为iterator的类型，而这种类型支持（概念上的）迭代器的各种操作。

每种容器都定义了一对命名为begin和end的函数，用于返回迭代器

begin返回的迭代器指向第一个元素：

vector<int>::iterator iter = ivec.begin();

由end操作返回的迭代器指向vector的“末端元素的下一个”，超出末端迭代器（off-the-end iterator），指向了一个不存在的元素。

由end操作返回的迭代器并不指向vector中任何实际的元素

迭代器类型使用解引用操作符（\*操作符）来访问迭代器所指向的元素

\*iter = 0;

自增操作符

比较两个迭代器 ==，!=

for(vector<int>::size\_type ix = 0; ix != ivec.size(); ++ix)

ivec[ix] = 0;