# C++STL之vector的说明及使用

## **（一）vector说明：**

vector向量是一种顺序行容器。相当于数组，但是其大小可以死不预先指定给，并且自动扩展。他可以像数组一样被操作，通过使用[]符号。因此可以将vector变量看做动态数组。

在创建vector变量后，它会自动在内存中分配一块连续的内存空间来保存数据，初始内存空间大小可以预先指定，也可以由vector们默认指定大小。当存储的数据超过分配的空间时，vector会重新分配一块内存，但是这样的分配很耗时，在重新分配空间是它的步骤如下:

(1)vector 会申请一块更大的内存块；

(2)将原来的数据拷贝到新的内存块中；

(3)销毁掉原内存块中的对象（调用对象的析构函数）；

(4)将原来的内存空间释放掉。

当vector保存的数据量很大时，如果此时进行插入数据导致需要更大的空间来保存这些数据量，那么将会大大的影响程序运行的效率，所以我们应该合理地使用vector。

## **（二）定义vector对象的方式：**

### **（1）初始化vector对象的方式：**

vector<T> v1; //未指定初始化方式，因此采用默认方式初始化v1，即长度为0的空容器；

vector<T> v2; //定义对象v2，作为v1的副本；

vector<T3> v3(n,i);//定义对象v3，它包含n个数值为i的元素；

vector<T> v4(n); //定义对象v4，v4中包含n个元素，每个元素的值均是0；

其中：T可以为int，float，char，char\*（存放字符串或字符指针）。一般用于int动态数值时，定义vector<T> v1，用v1来保存数值元素。

### **（2）vector的常用函数：只需要包含#include<vector>即可**

empty():判断vector向量是否为为空，为空时返回真，否则返回假；

begin():返回向量（数组）的首元素的地址；

end():返回向量（数组）的末尾元素的下一个元素（不存在的元素）的地址；

clear():清空向量；

front():返回向量（数组）的第一个元素的数据；

back():返回向量（数组）的最后一个元素的数据；

size():返回向量中元素的个数（即数组长度）

push\_back(数据i):将数据插入到向量的尾部；

pop\_back():删除向量尾部的数据

....

非常重要的两个操作：

**insert() 函数有以下三种用法:**

**1、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**

**2、在指定位置loc前插入num个值为val的元素**

**3、在指定位置loc前插入区间[start, end)的所有元素**

**vec.insert(vec.begin()+i,a);在第i+1个元素前面插入a;**

使用方法：对象调用成员函数采用符号“.”，如v1.push\_back(1)；

### **（3）遍历vector的方式：**

vecor向量支持2中方式遍历，因为可以认为vector是一个动态数组，所以可以使用数组下标的方式；或是使用vector的迭代器（vector的迭代器的具体使用参考文档《C++之迭代器Iterator》）****。****

## **#include <iostream>**

## **#include <vector>**

## **using namespace std;**

## **int main(void)**

## **{**

## **vector<int> vec;**

## **vec.push\_back(1);**

## **vec.push\_back(2);**

## **vec.push\_back(3);**

## **vec.push\_back(4);**

## **vec.push\_back(5);**

## **cout << "向量的大小：" << vec.size() << endl;**

## **// 数组下标方式遍历vector**

## **for (int i = 0; i < vec.size(); i++)**

## **cout << vec[i] << " ";**

## **cout << endl;**

## **// 迭代器方式遍历vector，注意：如果向量是常向量，如const vector<int> vec，**

## **//那么迭代器在定义是也需要定义为常量：vector<int>::const iterator itor = vec.begin();**

## **vector<int>::iterator itor = vec.begin();**

## **for (; itor != vec.end(); itor++)**

## **cout << \*itor << " ";**

## **cout << endl;**

## **return 0;**

## **}**

## **（三）vector的特点：**

(1) 指定一块如同数组一样的连续存储，但空间可以动态扩展。即它可以像数组一样数组一样被操作，通过[]符号。可以进行动态操作，通常体现在push\_back() pop\_back()；

(2) 随机访问方便，它像数组一样被访问，即支持[ ] 操作符和vector.at()；

(3) 节省空间，因为它是连续存储，在存储数据的区域都是没有被浪费的，但是要明确一点vector 大多情况下并不是满存的，在未存储的区域实际是浪费的；

(4) 在内部进行插入、删除操作效率非常低，这样的操作基本上是被禁止的。Vector 被设计成只能在后端进行追加和删除操作，其原因是vector 内部的实现是按照顺序表的原理；

(5) 只能在vector 的最后进行push 和pop ，不能在vector 的头进行push 和pop ；

(6) 当动态添加的数据超过vector 默认分配的大小时要进行内存的重新分配、拷贝与释放，这个操作非常消耗性能。 所以要vector 达到最优的性能，最好在创建vector 时就指定其空间大小