

系统架构师 少儿编程 云计算..

广告

使用list

并且关心两端数据的插入和删除，则应使用deque

1 少儿编程 2 python培训机 3 儿童编程 广告

c、若需要随机插入/删除（不仅仅在两端），则选择list

d、只有需要在首端进行插入/删除操作的时候，还要兼顾随机访问效率，才选择deque，否则都选择vector。

e、若既需要随机插入/删除，又需要随机访问，则需要在vector与list间做个折中-deque。

f、当要存储的是大型负责类对象时，list要优于vector；当然这时候也可以用vector来存储指向对象的指针，同样会取得较高的效率，但是指针的维护非常容易出错，因此不推荐使用。

问题一：list和vector的区别：

- (1) vector为存储的对象分配一块连续的地址空间，随机访问效率很高。但是插入和删除需要移动大量的数据，效率较低。尤其当vector中存储的对象较大，或者构造函数复杂，则在对现有的元素进行拷贝的时候会执行拷贝构造函数。
- (2) list中的对象是离散的，随机访问需要遍历整个链表，访问效率比vector低。但是在list中插入元素，尤其在首尾插入，效率很高，只需要改变元素的指针。
- (3) vector是单向的，而list是双向的；
- (4) 向量中的iterator在使用后就释放了，但是链表list不同，它的迭代器在使用后还可以继续用；链表特有的；

使用原则：

- (1) 如果需要高效的随机存取，而不在乎插入和删除的效率，使用vector；
- (2) 如果需要大量高效的删除插入，而不在乎存取时间，则使用list；
- (3) 如果需要搞笑的随机存取，还要大量的首尾的插入删除则建议使用deque，它是list和vector的折中；

问题二：常量容器const

const vector<int> vec(10);//这个容器里capacity和size和值都是不能改变的，const修饰的是vector；

迭代器：const vector<int>::const_iterator ite; //常量迭代器；

注：const vector <int> vec(10) —— 与const int a[10]是一回事，意思是vec只有10个元素，不能增加了，里面的元素也是不能变化的

```
1 vector<int> a(10);
2 const vector<int> b(10);
3 a[1]=10;//正确
4 b[1]=10;//错误
5 a.resize(20);//正确
6 b.resize(20);//错误
```

```
1 vector <const int> vec(10); //目前没有这种用法;这样写后也是当作vector <int>vec来用的;
2 关于vector<const int> , 在GCC下是没有这种用法的,编译不过
3 在VS2008是把它当作vector<int>这种类型来处理的;
```

问题三：capacity V.S size

- a、capacity是容器需要增长之前，能够盛的元素总数；只有连续存储的容器才有capacity的概念（例如vector，deque，string），list不需要capacity。
- b、size是容器当前存储的元素的数目。
- c、vector默认的容量初始值，以及增长规则是依赖于编译器的。

问题四：用vector存储自定义类对象时，自定义类对象须满足：

- a、有可供调用的无参构造函数（默认的或自定义的）；
- b、有可用的拷贝赋值函数（默认的或自定义的）

问题五：迭代器iterator

- a、vector与deque的迭代器支持算术运算，list的迭代器只能进行++/--操作，不支持普通的算术运算。
- b.向量中的iterator在使用后就释放了，但是链表list不同，它的迭代器在使用后还可以继续用；链表特有的；

```
1 ite=find(vec.begin(),vec.end(),88);
2 vec.insert(ite,2,77); //迭代器标记的位置前，插入数据;
3 cout<<*ite<<endl; //会崩溃，因为迭代器在使用后就释放了，*ite的时候就找不到它的地址了;
```

vector代码实例：

```
1 #include <iostream>
```



```
2 | using namespace std;
3 | #include <vector>    //向量的头文件；
4 | #include <algorithm> //算法的头文件；
5 | int main()
6 | {
7 |     vector<int> vec(5,8);
8 |     //--类型是vector<int>, 该容器向量中含有5个int类型的数值8, 变量名为vec。
9 |     //vector是一个类模板 (class template) ,所以必须要声明其类型, int, 一个容器中所有的对象必须是同一种类型；
10 |    // 定义一个容器对象；直接构造出一个数组；用法和数组一样；
11 |    //
12 |        for(int i=0;i<vec.size();i++)    //size()是指容器里当前有多少个使用的元素；
13 |        {
14 |            cout<<vec[i]<<" ";
15 |        }
16 |        cout<<endl<<vec.size()<<" "<<vec.capacity()<<endl;    //得到容器里用的多少个空间, 和总共的大小；
17 |        vector<int>::iterator ite;    //定义了一个向量的迭代器；相当于定义了一个指针；
18 |    for(ite=vec.begin();ite!=vec.end();ite++)    //得到开始、结束
19 |    {
20 |        cout<<*ite <<" ";    //迭代器返回的是引用；
21 |    }
22 |    cout<<endl;
23 |    //在尾部插入；
24 |    vec.push_back(9);    //VS6.0扩充的空间是两倍；在VS2005扩充的空间是1.5倍；
25 |    for(ite=vec.begin();ite!=vec.end();ite++)    //得到开始、结束
26 |    {
27 |        cout<<*ite <<" ";
28 |    }
29 |    cout<<endl<<vec.size()<<" "<<vec.capacity()<<endl;
30 |
31 |    //尾部删除;容量没变【capacitty】, 但是使用空间减少一个；容量一旦增加就不会减小；
32 |    vec.pop_back();
33 |    for(ite=vec.begin();ite!=vec.end();ite++)    //得到开始、结束
34 |    {
35 |        cout<<*ite <<" ";
36 |    }
37 |    cout<<endl<<vec.size()<<" "<<vec.capacity()<<endl;
38 |
39 |    vec.push_back(88);
40 |    vec.push_back(99); //容量刚好够；
41 |
42 |    for(ite=vec.begin();ite!=vec.end();ite++)    //得到开始、结束
43 |    {
44 |        cout<<*ite <<" ";
45 |    }
46 |    cout<<endl<<vec.size()<<" "<<vec.capacity()<<endl;
47 |
48 |    ite = find(vec.begin(),vec.end(),88);    //查找这个元素；
49 |    vec.erase(ite);    //利用迭代器指针删除这个元素；
50 |    for(int i=0;i<vec.size();i++)    //size()是指容器里当前有多少个使用的元素；
51 |    {
52 |        cout<<vec[i]<<" ";
53 |    }
54 |    cout<<endl<<vec.size()<<" "<<vec.capacity()<<endl;    //得到容器里用的多少个空间, 和总共的大小；
55 |
56 |    vec.clear(); //只是清除了数据, 没有回收空间, 空间的等到对象的生命周期结束时回收；
57 |    //使用空间为0, 但是容量的空间还在, 只有在调用析构函数的时候空间才会回收；
58 |
59 |    for(int i=0;i<vec.size();i++)    //size()是指容器里当前有多少个使用的元素；
60 |    {
61 |        cout<<vec[i]<<" ";
62 |    }
63 |    cout<<endl<<vec.size()<<" "<<vec.capacity()<<endl;
64 |
65 |    ite=find(vec.begin(),vec.end(),88);
66 |    vec.insert(ite,2,77);    //迭代器标记的位置前, 插入数据；
67 |
68 |    <span style="color:#ff0000;">//cout<<*ite<<endl;    //会崩溃, 因为迭代器在使用后就释放了, *ite的时候就找不到它的地址了；
```

```
69 //和向量的用法一样，但是链表list不同，它的迭代器在使用后还可以继续用；链表特有的；</span>
70
71 for(int i=0;i<vec.size();i++)
72 {
73     cout<<vec[i]<<"  ";
74 }
75 cout<<endl<<vec.size()<<" "<<vec.capacity()<<endl;
76
77 system("pause");
78 return 0;
79 }
```

运行结果：

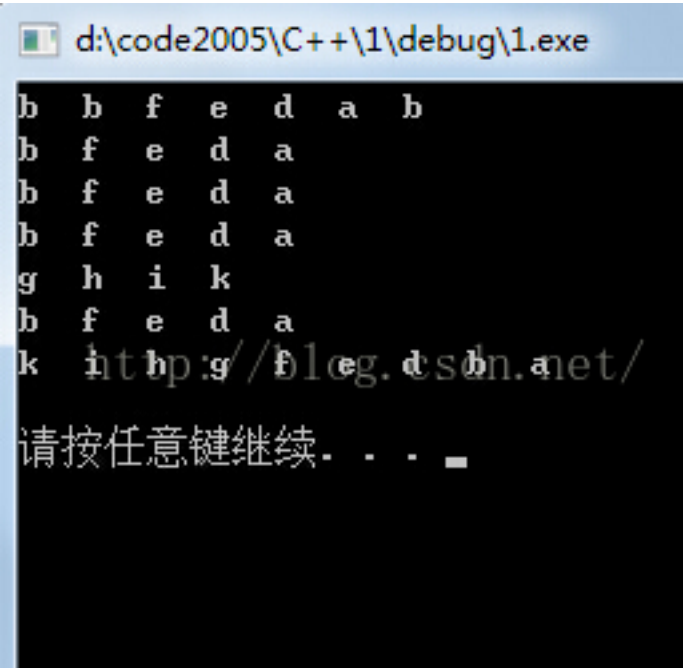


list代码示例：

```
1 #include<iostream>
2 #include <list>
3 #include <algorithm>
4 using namespace std;
5 int main()
6 {
7     list<char> lit;
8     //用法和向量一样,
9     //list是一个类模板, template, char是链表里对象的类型, lit是创建的一个对象;
10    //链表可以再头尾两端插入,是双向的;
11
12    lit.push_back('a');
13    lit.push_back('b');
14    lit.push_front('d');
15    lit.push_front('e');
16    lit.push_front('f');
17    lit.push_front('b');
18    lit.push_front('b');
19
20    list<char>::iterator it; //定义一个list的迭代器, 类似一个纸箱链表的指针, 但是比一般的指针好用, 里面用到了好多重载操作;
21    list<char>::iterator it1;
22    list<char>::iterator it2;
23    for(it=lit.begin();it!=lit.end();it++)
24    {
25        cout<<*it<<" ";
26    }
27    cout<<endl;
28    //-----链表可以从两端删除-----
29    lit.pop_back();
30    lit.pop_front();
31    for(it=lit.begin();it!=lit.end();it++)
32    {
33        cout<<*it<<" ";
34    }
35    cout<<endl;
```

```
36 //-----删除所有的a----- 37 | //lit.remove('a'); //删除所有的a;
37
38
39 for(it=lit.begin();it!=lit.end();it++)
40 {
41     cout<<*it<<" ";
42 }
43 cout<<endl;
44 //-----移除连续且相同的a, 只剩下一个;-----
45 lit.unique(); //移除连续且相同的a, 只剩下一个;
46
47 for(it=lit.begin();it!=lit.end();it++)
48 {
49     cout<<*it<<" ";
50 }
51 cout<<endl;
52 list<char> lit1;
53 lit1.push_back('g');
54 lit1.push_back('h');
55 lit1.push_back('i');
56 lit1.push_back('k');
57 for(it1=lit1.begin();it1!=lit1.end();it1++)
58 {
59     cout<<*it1<<" ";
60 }
61 cout<<endl;
62 //-----将一个链表插入到另一个链表-----
63 it1=find(lit.begin(),lit.end(),'f'); //先的找到要插入的位置, 在该位置的前一个插入;
64 ///lit.splice(it1,lit1); //将第二个链表插入到第一个链表中; 合并后的链表就没了, 因为传的是&;
65 for(it=lit.begin();it!=lit.end();it++)
66 {
67     cout<<*it<<" ";
68 }
69 cout<<endl;
70 //-----在链表lit中的it前插入lit1中的一个元素it1; 在f之前插入k-----
71 //-----拿下来之后那个元素就没有了-----
72 it=find(lit.begin(),lit.end(),'f');
73 it1=find(lit1.begin(),lit1.end(),'k');
74 lit.splice(it,lit1,it1);
75 //-----把链表中的一段插入到另一个链表中-----
76 //把链表lit1中的[it-----it1)段的字符插入到lit的it2指针前;
77 it=find(lit1.begin(),lit1.end(),'h');
78 it1=find(lit1.begin(),lit1.end(),'k');
79 it2=find(lit.begin(),lit.end(),'f');
80 lit.splice(it2,lit1,it,it1);
81 // ----void merge(list& x); //将x合并到*this 身上。两个lists 的内容都必须先经过递增归并排序。
82 lit.sort(); //对两个排序进行归并排序;
83 lit1.sort();
84 lit.merge(lit1);
85 //-----将list里的数据倒序排列-----
86 lit.reverse();
87 for(it=lit.begin();it!=lit.end();it++)
88 {
89     cout<<*it<<" ";
90 }
91 cout<<endl;
92 for(it1=lit1.begin();it1!=lit1.end();it1++)
93 {
94     cout<<*it1<<" ";
95 }
96 cout<<endl;
97 system("pause");
98 return 0;
99 }
100
101
```


运行结果：




文章标签： [iterator](#) [list](#) [deque](#) [vector](#) [容器](#)


个人分类： [C++ PP\(Edit 6\)](#)


所属专栏： [C++基础](#)



想对作者说点什么？ [我来说一句](#)

 一只笨鸟 2018-03-06 14:38:33 #4楼
写的很棒，以后参考使用。

 cccvvv1234 2018-01-02 22:49:51 #3楼
干的【漂亮】

 know_heng 2017-07-10 16:49:43 #2楼
写的很好

[查看 4 条热评](#)

C++deque双端队列

调用头文件： `#include using namespace std;` 详细用法(部分)： `deque k;` ----- 定义一个deque的变量(定义时已经初始化...

 Jaihk662 2016-08-09 11:49:10 阅读数： 3351

【C++】STL常用容器总结之五：双端队列deque

所谓的deque是”double ended queue”的缩写，双端队列不论在尾部或头部插入元素，都十分迅速。而在中间插入元素则会比较费时，因为必须移动中间其他的元素。双端队列

Linux下Shell从入门到精通完整版

Linux | wanghui_777



Java架构师之路——Java中高阶知识精讲

Java | wanghui_777



C++ queue 和 deque的区别

先说queue，queue支持5中基本的操作： empty,

 bemachine 2014-08-12 13:07:19 阅读数：9335

C++中的deque总结

Deque双端队列容器 一、基本原理 deque的元素数据采用分块的线性结构进行存储，如图所示。deque分成若干线性存储块，称为deque块。块的大小一般为512个字节，元素的数据类型所...

 sinat_36165006 2017-02-19 12:09:43 阅读数：1607

【C++ STL学习之三】容器deque深入学习

C++ STL容器deque和vector很类似，也是采用动态数组来管理元素。使用deque之前需包含头文件： #include 它是定义在命名空间std内的一个class temp...

 xiajun07061225 2012-04-10 08:55:17 阅读数：17000

C++/C++11中std::deque的使用

C++/C++11中std::deque的使用

 fengbingchun 2017-05-25 21:27:08 阅读数：3446

男人补肾方法！教你1个方法，做回雄风男人！

星暹 · 顶新




男人补肾方法！教你1个方法，做回雄风男人！

星暹 · 顶新



c++ queue 和 deque

queue操作： queueq; 创建一个int型空队列q q.empty(); 判断队列是否为空，为空返回true q.push(s); 将变量s从队尾入队 q.pop(); ...

 qq_36238595 2017-02-07 08:26:30 阅读数：1372

deque

deque 双端队列容器 deque 双端队列容器 (double-ended queue) 与 vector 非常相似，不仅可在尾部插入和删除元素，还可以在头部插入和删除，算法的时间...

 ZWY18064117182 2016-09-02 22:11:11 阅读数：2840

C++ STL之deque详解

deque容器deque容器是C++ STL中的内容。deque与vector类似，支持随机访问和快速插入删除。deque还支持从开始端加入数据:push_front()构造函数deque d;//创...

 Area_52 2015-06-11 15:12:03 阅读数：2268

C++ Deque容器的使用方法

Deque容器的使用方法 容器deque和vector非常相似，操作函数基本一致。它采用动态数组来管理元素，提供随机存取，可以在头尾两端进行快速安插和删除元素操作。特别

没有更多推荐了，[返回首页](#)

个人资料



尘虚缘_KY

关注

原创

粉丝

喜欢

评论

252

117

29

69

等级：

博客 6

访问：

39万+

积分：

6910

排名：

4461

勋章：







整容脸多少钱



最新文章



少儿编程



种头发危害



Mac上安装gradle和maven



jetty运行配置ERROR

博主专栏

整容脸多少钱

儿童编程



C专家编程

阅读量：16081

13 篇



剑指offer

阅读量：74272

70 篇



C++基础

阅读量：84943

39 篇



编程之美

阅读量：12843

8 篇

Keep going~

时间不会因为你的迷茫而停留；
生活不会因为你的惆怅而改变；
即使此刻：
不知有多少人仍在不停的翻动着书页.....

个人分类	
C 基础	22篇
Linux 基础	26篇
数据结构与算法	27篇
C++ PP(Edit 6)	45篇
Java 基础	6篇
展开	

归档	
2017年6月	2篇
2017年5月	4篇
2016年8月	4篇
2016年7月	13篇
2016年6月	70篇
展开	

最新评论	
C++的初始化列表(Initili...	MirabelleZWH： 一下子就看懂了
C++的初始化列表(Initili...	MirabelleZWH： 写的非常棒
C++的初始化列表(Initili...	MirabelleZWH： 请继续加油
c语言实现动态心型代码	f_zyj： 然而并不是心，一堆大大小小的异或符
哈希表在Top-k问题中的应用--...	yulutian： 这个归并排序也需要时间复杂度的