

vector容器用法详解

vector类称作向量类，它实现了动态数组，用于元素数量变化的对象数组。像数组一样，vector类也用从0开始的下标表示元素的位置；但和数组不同的是，当vector对象创建后，数组的元素个数会随着vector对象元素个数的增大和缩小而自动变化。

vector类常用的函数如下所示：

1.构造函数

- vector():创建一个空vector
- vector(int nSize):创建一个vector,元素个数为nSize
- vector(int nSize,const t& t):创建一个vector，元素个数为nSize,且值均为t
- vector(const vector&):复制构造函数
- vector(begin,end):复制[begin,end)区间内另一个数组的元素到vector中

2.增加函数

- void push_back(const T& x):向量尾部增加一个元素X
- iterator insert(iterator it,const T& x):向量中迭代器指向元素前增加一个元素x
- iterator insert(iterator it,int n,const T& x):向量中迭代器指向元素前增加n个相同的元素x
- iterator insert(iterator it,const_iterator first,const_iterator last):向量中迭代器指向元素前插入另一个相同类型向量的[first,last)间的数据

3.删除函数

- iterator erase(iterator it):删除向量中迭代器指向元素
- iterator erase(iterator first,iterator last):删除向量中[first,last)中元素
- void pop_back():删除向量中最后一个元素
- void clear():清空向量中所有元素

4.遍历函数

2019年3月						
日	一	二	三	四	五	六
24	25	26	27	28	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

导航

[博客园](#)
[首页](#)
[新随笔](#)
[联系](#)
[订阅](#) XML
[管理](#)

统计

随笔 - 145
文章 - 0
评论 - 1
引用 - 0

公告

昵称： 蒲公英110
园龄： 3年11个月
粉丝： 16
关注： 0
[+加关注](#)

搜索

常用链接

[我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

随笔分类

[C++\(14\)](#)
[C++多线程\(23\)](#)
[C关键字\(5\)](#)
[oracle](#)
[TCP/IP\(1\)](#)
[VS2013软件相关\(4\)](#)
[程序编译相关](#)

- `reference at(int pos)`:返回pos位置元素的引用
- `reference front()`:返回首元素的引用
- `reference back()`:返回尾元素的引用
- `iterator begin()`:返回向量头指针, 指向第一个元素
- `iterator end()`:返回向量尾指针, 指向向量最后一个元素的下一个位置
- `reverse_iterator rbegin()`:反向迭代器, 指向最后一个元素
- `reverse_iterator rend()`:反向迭代器, 指向第一个元素之前的位置

5.判断函数

- `bool empty() const`:判断向量是否为空, 若为空, 则向量中无元素

6.大小函数

- `int size() const`:返回向量中元素的个数
- `int capacity() const`:返回当前向量所能容纳的最大元素值
- `int max_size() const`:返回最大可允许的vector元素数量值

7.其他函数

- `void swap(vector&)`:交换两个同类型向量的数据
- `void assign(int n,const T& x)`:设置向量中第n个元素的值为x
- `void assign(const_iterator first,const_iterator last)`:向量中[first,last)中元素设置成当前向量元素

示例:

1.初始化示例

[cpp] view plain copy

```
1. #include "stdafx.h"
2. #include<iostream>
3. #include<vector>
4.
5. using namespace std;
```

程序常见错误整理(3)

传统文化(11)
工作和生活(4)
基本概念系列(1)
面试知识点(4)
软件调试相关(2)
数据库(3)
数学相关(1)
统计学(2)
硬件相关(2)
怎么思考问题(1)
找工作经验
智力题(1)

随笔档案

2017年1月 (1)
2016年12月 (2)
2016年11月 (16)
2016年10月 (89)
2016年9月 (26)
2016年8月 (11)

最新评论

1. Re:热门智力题 过桥问题和倒水问题
这位老师讲的真好!!
--马扁钱先生

阅读排行榜

1. vector容器用法详解(60431)
2. vector中erase用法注意事项(10564)
3. 对多维向量vector<vector<int>>vec进行操作(7613)
4. Oracle数据库中scott用户不存在的解决方法(6647)
5. 关于c++显示调用析构函数的陷阱(5922)

评论排行榜

1. 热门智力题 过桥问题和倒水问题(1)

推荐排行榜

1. vector容器用法详解(4)
2. 对多维向量vector<vector<int>>vec进行操作(1)
3. 热门智力题 过桥问题和倒水问题(1)
4. 关于c++显示调用析构函数的陷阱(1)
5. 虚函数、纯虚函数和接口的实用方法和意义(1)

```
6.
7.  class A
8.  {
9.      //空类
10. };
11. int _tmain(int argc, _TCHAR* argv[])
12. {
13.
14.     //int型vector
15.     vector<int> vecInt;
16.
17.     //float型vector
18.     vector<float> vecFloat;
19.
20.     //自定义类型，保存类A的vector
21.     vector<A> vecA;
22.
23.     //自定义类型，保存指向类A的指针的vector
24.     vector<A*> vecPointA;
25.
26.     return 0;
27. }
```

[cpp] view plain copy

```
1.  // vectorsample.cpp : 定义控制台应用程序的入口点。
2.  //
3.
4.  #include "stdafx.h"
5.  #include<iostream>
6.  #include<vector>
7.
8.  using namespace std;
9.
10. class A
11. {
12.     //空类
13. };
14. int _tmain(int argc, _TCHAR* argv[])
15. {
16.
17.     //int型vector,包含3个元素
18.     vector<int> vecIntA(3);
19.
20.     //int型vector,包含3个元素且每个元素都是9
21.     vector<int> vecIntB(3,9);
22.
23.     //复制vecIntB到vecIntC
24.     vector<int> vecIntC(vecIntB);
25.
26.     int iArray[]={2,4,6};
27.     //创建vecIntD
```

```

28.     vector<int> vecIntD(iArray,iArray+3);
29.
30.     //打印vectorA,此处也可以用下面注释内的代码来输出vector
    中的数据
31.     /*for(int i=0;i<vecIntA.size();i++)
32.     {
33.         cout<<vecIntA[i]<<"    ";
34.     }*/
35.
36.     cout<<"vecIntA:"<<endl;
37.     for(vector<int>::iterator it = vecIntA.begin();it!=
    vecIntA.end();it++)
38.     {
39.         cout<<*it<<"    ";
40.     }
41.     cout<<endl;
42.
43.     //打印vecIntB
44.     cout<<"VecIntB:"<<endl;
45.     for(vector<int>::iterator it = vecIntB.begin() ;it!
    =vecIntB.end();it++)
46.     {
47.         cout<<*it<<"    ";
48.     }
49.     cout<<endl;
50.
51.     //打印vecIntC
52.     cout<<"VecIntB:"<<endl;
53.     for(vector<int>::iterator it = vecIntC.begin() ;it!
    =vecIntC.end();it++)
54.     {
55.         cout<<*it<<"    ";
56.     }
57.     cout<<endl;
58.
59.     //打印vecIntD
60.     cout<<"vecIntD:"<<endl;
61.     for(vector<int>::iterator it = vecIntD.begin() ;it!
    =vecIntD.end();it++)
62.     {
63.         cout<<*it<<"    ";
64.     }
65.     cout<<endl;
66.     return 0;
67. }

```

程序的运行结果如下：

上面的代码用了4种方法建立vector并对其初始化

2.增加及获得元素示例：

[\[cpp\]](#) [view](#) [plain](#) [copy](#)

```
1. // vectorsample.cpp : 定义控制台应用程序的入口点。
2. //
3.
4. #include "stdafx.h"
5. #include<iostream>
6. #include<vector>
7.
8. using namespace std;
9.
10.
11. int _tmain(int argc, _TCHAR* argv[])
12. {
13.
14.     //int型vector, 包含3个元素
15.     vector<int> vecIntA;
16.
17.     //插入1 2 3
18.     vecIntA.push_back(1);
19.     vecIntA.push_back(2);
20.     vecIntA.push_back(3);
21.
22.     int nSize = vecIntA.size();
23.
24.     cout<<"vecIntA:"<<endl;
25.
26.     //打印vectorA, 方法一:
27.     for(int i=0;i<nSize;i++)
28.     {
29.         cout<<vecIntA[i]<<" ";
30.     }
31.     cout<<endl;
32.
33.     //打印vectorA, 方法二:
34.     for(int i=0;i<nSize;i++)
35.     {
36.         cout<<vecIntA.at(i)<<" ";
37.     }
38.     cout<<endl;
39.
40.     //打印vectorA, 方法三:
41.     for(vector<int>::iterator it = vecIntA.begin();it!=
vecIntA.end();it++)
42.     {
43.         cout<<*it<<" ";
44.     }
45.     cout<<endl;
46.
47.     return 0;
48. }
```

上述代码对一个整形向量类进行操作，先定义一个整形元素向量

类，然后插入3个值，并用3种不同的方法输出，程序运行结果如下：

[cpp] view plain copy

```
1. // vectorsample.cpp : 定义控制台应用程序的入口点。
2. //
3.
4. #include "stdafx.h"
5. #include<iostream>
6. #include<vector>
7.
8. using namespace std;
9.
10. class A
11. {
12. public:
13.     int n;
14. public:
15.     A(int n)
16.     {
17.         this->n = n;
18.     }
19. };
20.
21. int _tmain(int argc, _TCHAR* argv[])
22. {
23.
24.     //int型vector,包含3个元素
25.     vector<A> vecClassA;
26.
27.     A a1(1);
28.     A a2(2);
29.     A a3(3);
30.
31.     //插入1 2 3
32.     vecClassA.push_back(a1);
33.     vecClassA.push_back(a2);
34.     vecClassA.push_back(a3);
35.
36.
37.     int nSize = vecClassA.size();
38.
39.     cout<<"vecClassA:"<<endl;
40.
41.     //打印vecClassA,方法一:
42.     for(int i=0;i<nSize;i++)
43.     {
44.         cout<<vecClassA[i].n<<" ";
45.     }
```

```

46.         cout<<endl;
47.
48.         //打印vecClassA,方法二:
49.         for(int i=0;i<nSize;i++)
50.         {
51.             cout<<vecClassA.at(i).n<<"    ";
52.         }
53.         cout<<endl;
54.
55.         //打印vecClassA,方法三:
56.         for(vector<A>::iterator it = vecClassA.begin();it!=
vecClassA.end();it++)
57.         {
58.             cout<<(*it).n<<"    ";
59.         }
60.         cout<<endl;
61.
62.         return 0;
63.     }

```

上述代码通过定义元素为类的向量，通过插入3个初始化的类，并通过3种方法输出，运行结果如下：

[cpp] view plain copy

```

1.     // vectorsample.cpp : 定义控制台应用程序的入口点。
2.     //
3.
4.     #include "stdafx.h"
5.     #include<iostream>
6.     #include<vector>
7.
8.     using namespace std;
9.
10.    class A
11.    {
12.    public:
13.        int n;
14.    public:
15.        A(int n)
16.        {
17.            this->n = n;
18.        }
19.    };
20.
21.    int _tmain(int argc, _TCHAR* argv[])
22.    {
23.

```

```
24. //int型vector, 包含3个元素
25. vector<A*> vecClassA;
26.
27. A *a1 = new A(1);
28. A *a2 = new A(2);
29. A *a3 = new A(3);
30.
31. //插入1 2 3
32. vecClassA.push_back(a1);
33. vecClassA.push_back(a2);
34. vecClassA.push_back(a3);
35.
36.
37. int nSize = vecClassA.size();
38.
39. cout<<"vecClassA:"<<endl;
40.
41. //打印vecClassA, 方法一:
42. for(int i=0;i<nSize;i++)
43. {
44.     cout<<vecClassA[i]->n<<"\t";
45. }
46. cout<<endl;
47.
48. //打印vecClassA, 方法二:
49. for(int i=0;i<nSize;i++)
50. {
51.     cout<<vecClassA.at(i)->n<<"\t";
52. }
53. cout<<endl;
54.
55. //打印vecClassA, 方法三:
56. for(vector<A*>::iterator it = vecClassA.begin();it!=vecClassA.end();it++)
57. {
58.     cout<<(**it).n<<"\t";
59. }
60. cout<<endl;
61. delete a1; delete a2;delete a3;
62. return 0;
63. }
```

上述代码通过定义元素为类指针的向量，通过插入3个初始化的类指针，并通过3种方法输出指针指向的类，运行结果如下：

3.修改元素示例

修改元素的方法主要有三种：1.通过数组修改，2.通过引用修改，3.通过迭代器修改

 [cpp] view plain copy


```
1. // vectorsample.cpp : 定义控制台应用程序的入口点。
2. //
3.
4. #include "stdafx.h"
5. #include<iostream>
6. #include<vector>
7.
8. using namespace std;
9.
10.
11. int _tmain(int argc, _TCHAR* argv[])
12. {
13.
14.     //int型vector,包含3个元素
15.     vector<int> vecIntA;
16.
17.     //插入1 2 3
18.     vecIntA.push_back(1);
19.     vecIntA.push_back(2);
20.     vecIntA.push_back(3);
21.
22.     int nSize = vecIntA.size();
23.
24.     //通过引用修改vector
25.     cout<<"通过数组修改, 第二个元素为8: "<<endl;
26.     vecIntA[1]=8;
27.
28.     cout<<"vecIntA:"<<endl;
29.     //打印vectorA
30.     for(vector<int>::iterator it = vecIntA.begin();it!=
vecIntA.end();it++)
31.     {
32.         cout<<*it<<" ";
33.     }
34.     cout<<endl;
35.
36.     //通过引用修改vector
37.     cout<<"通过引用修改, 第二个元素为18: "<<endl;
38.     int &m = vecIntA.at(1);
39.     m=18;
40.
41.     cout<<"vecIntA:"<<endl;
42.     //打印vectorA
43.     for(vector<int>::iterator it = vecIntA.begin();it!=
vecIntA.end();it++)
44.     {
45.         cout<<*it<<" ";
46.     }
47.     cout<<endl;
48.
49.     //通过迭代器修改vector
50.     cout<<"通过迭代器修改, 第二个元素为28"<<endl;
51.     vector<int>::iterator itr = vecIntA.begin()+1;
```

```
52.     *itr = 28;
53.
54.     cout<<"vecIntA:"<<endl;
55.     //打印vectorA
56.     for(vector<int>::iterator it = vecIntA.begin();it!=
vecIntA.end();it++)
57.     {
58.         cout<<*it<<"    ";
59.     }
60.     cout<<endl;
61.
62.     return 0;
63. }
```

程序运行结果如下：

4.删除向量示例

删除向量主要通过erase和pop_back，示例代码如下

[cpp] view plain copy

```
1.  // vectorsample.cpp : 定义控制台应用程序的入口点。
2.  //
3.
4.  #include "stdafx.h"
5.  #include<iostream>
6.  #include<vector>
7.
8.  using namespace std;
9.
10.
11. int _tmain(int argc, _TCHAR* argv[])
12. {
13.
14.     //int型vector,包含3个元素
15.     vector<int> vecIntA;
16.
17.     //循环插入1 到10
18.     for(int i=1;i<=10;i++)
19.     {
20.         vecIntA.push_back(i);
21.     }
22.
23.     vecIntA.erase(vecIntA.begin()+4);
24.
25.     cout<<"删除第5个元素后的向量vecIntA:"<<endl;
26.     //打印vectorA
27.     for(vector<int>::iterator it = vecIntA.begin();it!=
vecIntA.end();it++)
28.     {
```

```
29.         cout<<*it<<"\t";
30.     }
31.     cout<<endl;
32.
33.     //删除第2-5个元素
34.     vecIntA.erase(vecIntA.begin()+1,vecIntA.begin()+5);
35.
36.     cout<<"删除第2-5个元素后的vecIntA:"<<endl;
37.     //打印vectorA
38.     for(vector<int>::iterator it = vecIntA.begin();it!=
vecIntA.end();it++)
39.     {
40.         cout<<*it<<"\t";
41.     }
42.     cout<<endl;
43.
44.     //删除最后一个元素
45.     vecIntA.pop_back();
46.
47.     cout<<"删除最后一个元素后的vecIntA:"<<endl;
48.     //打印vectorA
49.     for(vector<int>::iterator it = vecIntA.begin();it!=
vecIntA.end();it++)
50.     {
51.         cout<<*it<<"\t";
52.     }
53.     cout<<endl;
54.
55.     return 0;
56. }
```

程序运行结果如下：

3.进一步理解vector，如下图所示：

当执行代码`vector<int> v(2,5)`时，在内存里建立了2个整形元素空间，值是5.当增加一个元素时，原有的空间由2个编程4个整形元素空间，并把元素1放入第3个整形空间，第4个空间作为预留空间。当增加元素2时，直接把值2放入第4个空间。当增加元素3时，由于原有向量中没有预留空间，则内存空间由4个变为8个整形空间，并把值放入第5个内存空间。

总之，扩大新元素时，如果超过当前的容量，则容量会自动扩充2倍，如果2倍容量仍不足，则继续扩大2倍。本图是直接在原空间基础上画的新增空间，其实要复杂的多，包括重新配置、元素移动、释放原始空间的过程。因此对vector容器而言，当增加新的元素时，有可能很快完成（直接存在预留空间中），有可能稍慢（扩容后再放新元素）；对修改元素值而言是较快的；对删除元素来

说，弱删除尾部元素较快，非尾部元素稍慢，因为牵涉到删除后的元素移动。

4.综合示例

[cpp] view plain copy

```
1. // vectorsample.cpp : 定义控制台应用程序的入口点。
2. //
3.
4. #include "stdafx.h"
5. #include<iostream>
6. #include<vector>
7. #include<string>
8. using namespace std;
9.
10. class Student
11. {
12. public:
13.     string m_strNO;
14.     string m_strName;
15.     string m_strSex;
16.     string m_strDate;
17. public:
18.     Student(string strNO,string strName,string strSex,string strDate)
19.     {
20.         m_strNO = strNO;
21.         m_strName = strName;
22.         m_strSex = strSex;
23.         m_strDate = strDate;
24.     }
25.     void Display()
26.     {
27.         cout<<m_strNO<<"\t";
28.         cout<<m_strName<<"\t";
29.         cout<<m_strSex<<"\t";
30.         cout<<m_strDate<<"\t";
31.     }
32. };
33.
34. class StudCollect
35. {
36.     vector<Student> m_vStud;
37. public:
38.     void Add(Student &s)
39.     {
40.         m_vStud.push_back(s);
41.     }
42.     Student* Find(string strNO)
43.     {
44.         bool bFind = false;
45.         int i;
46.         for(i = 0;i < m_vStud.size();i++)
```

```
47.         {
48.             Student& s = m_vStud.at(i);
49.             if(s.m_strNO == strNO)
50.             {
51.                 bFind = true;
52.                 break;
53.             }
54.         }
55.         Student *s = NULL;
56.         if(bFind)
57.             s = &m_vStud.at(i);
58.         return s;
59.     }
60. };
61.
62. int _tmain(int argc, _TCHAR* argv[])
63. {
64.     Student s1("1001", "zhangsan", "boy", "1988-10-
65.     10");
66.     Student s2("1002", "lisi", "boy", "1988-8-25");
67.     Student s3("1003", "wangwu", "boy", "1989-2-14");
68.
69.     StudCollect s;
70.     s.Add(s1);
71.     s.Add(s2);
72.     s.Add(s3);
73.
74.     Student *ps = s.Find("1002");
75.     if(ps)
76.         ps->Display();
77.     return 0;
78. }
```

代码运行实例如下：



蒲公英110

关注 - 0

粉丝 - 16

+加关注

4

0

« 上一篇: [对多维向量vector<vector<int> > vec进行操作](#)

» 下一篇: [vector中erase用法注意事项](#)

posted on 2016-10-03 23:08 [蒲公英110](#) 阅读(60431) 评论(0) [编辑](#) [收藏](#)

[刷新评论](#) [刷新页面](#) [返回顶部](#)