**一、栈（stack）**

引入头文件

[?](https://www.jb51.net/article/122462.htm)

|  |  |
| --- | --- |
| 1 | #include<stack> |

常用的方法

1. empty() 堆栈为空则返回真
2. pop() 移除栈顶元素
3. push() 在栈顶增加元素
4. size() 返回栈中元素数目
5. top() 返回栈顶元素

3.实例代码

[?](https://www.jb51.net/article/122462.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | #include<iostream>  #include<stack>  using namespace std;  int main(){    //创建栈 s    stack<int> s;    //将元素压入栈    for(int i=0;i<10;i++){      s.push(i);    }  //  if(!s.empty()){  //    cout<<"栈s不是空的"<<endl;  //  }    cout<<"栈s中元素的个数为："<<s.size()<<endl;    while(!s.empty()){      cout<<" "<<s.top();//获取栈顶元素      s.pop();//弹出栈顶元素    }    cout<<endl;    if(s.empty()){      cout<<"栈s现在为空"<<endl;    }    return 0;  } |

运行结果



**二、队列（queue）**

引入头文件

[?](https://www.jb51.net/article/122462.htm)

|  |  |
| --- | --- |
| 1 | #include<queue> |

常用方法

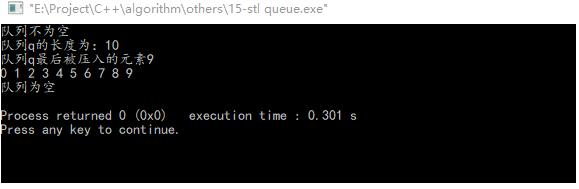
1. empty() 判断队列是否为空，返回类型为bool
2. size() 返回队列中元素的个数
3. front() 返回队列队首元素
4. back() 返回队列队尾元素
5. push(ele) 将元素ele插入到队尾
6. pop 队首元素出队

实例代码

[?](https://www.jb51.net/article/122462.htm)

|  |  |
| --- | --- |
| 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 | #include<iostream>  #include<queue>  using namespace std;  int main(){    int ele,num,last;    queue<int> q;    for(int i=0;i<10;i++){      q.push(i);    }    if(!q.empty()){      cout<<"队列不是空的"<<endl;    }    num=q.size();    cout<<"队列q的长度为："<<num<<endl;    last=q.back();    cout<<"队列q最后一个元素："<<last<<endl;    for(int j=0;j<q.size();j++){      ele=q.front();      cout<<ele<<" ";      q.pop();    }    cout<<endl;    if(q.empty()){      cout<<"队列是空的"<<endl;    }    return 0;  } |

运行结果



**三、向量(vector)**

引入头文件

[?](https://www.jb51.net/article/122462.htm)

|  |  |
| --- | --- |
| 1 | #include<vector> |

常用方法

1. c.clear() 移除容器中所有数据。
2. c.empty() 判断容器是否为空。
3. c.erase(pos) 删除pos位置的数据
4. c.erase(beg,end) 删除[beg,end)区间的数据
5. c.front() 传回第一个数据。
6. c.insert(pos,elem) 在pos位置插入一个elem拷贝
7. c.pop\_back() 删除最后一个数据。
8. c.push\_back(elem) 在尾部加入一个数据。
9. c.resize(num) 重新设置该容器的大小
10. c.size() 回容器中实际数据的个数。
11. c.begin() 返回指向容器第一个元素的迭代器
12. c.end() 返回指向容器最后一个元素的迭代器

代码实例

[?](https://www.jb51.net/article/122462.htm)

|  |  |
| --- | --- |
| 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 | #include<iostream>  #include<vector>  using namespace std;  int main(){    //向向量v中添加元素    vector<int> v;    for(int i=0;i<10;i++){      v.push\_back(i);    }    //从向量v中读取数据    for(int i=0;i<v.size();i++){      cout<<" "<<v[i];    }    cout<<endl;    //使用迭代器遍历数据 迭代器相当于指针    cout<<"使用迭代器获取数据"<<endl;    vector<int>::iterator iter;    for( iter = v.begin(); iter != v.end(); iter++ )    {     cout<<" "<< \*iter;    }    cout<<endl;    //删除数据    for(vector<int>::iterator it=v.begin();it!=v.end();){      v.erase(it);    }    if(v.empty()){      cout<<"向量v为空"<<endl;    }    return 0;  } |

运行结果

