山东大学 计算机科学与技术 学院

数据结构与算法 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201700130009 | 姓名： 张愈博 | | 班级： 计科17.3 |
| 实验题目：队列 | | | |
| 实验学时：2 | | 实验日期： 2018.11.11 | |
| 实验目的：  1、掌握队列结构的定义与实现。  2、掌握队列结构的使用。 | | | |
| 软件环境：  Windows 10  DEV C++ 5.11 | | | |
| 1. 实验内容（题目内容，输入要求，输出要求）   （1）创建队列类，采用链式描述；  （2）实现卡片游戏  假设桌上有一叠扑克牌，依次编号为 1-n（从最上面开始）。当至少还有两张的时候，可以进行操作：把第一张牌扔掉，然后把新的第一张放到整叠牌的最后。输入 n，输出每次要扔掉的牌，以及最后剩下的牌。   1. 数据结构与算法描述 （整体思路描述，所需要的数据结构与算法）   本实验划分为3个文件，arrayQueue.h 、扑克牌.h和main.cpp。  (1)在arrayQueue.h头文件中定义链式描述的队列类，其中包含方法判断是否空empty,求大小size,取队列首元素front，取队列尾元素back，从队首删除元素pop，从队尾添加元素push和输出队列元素output；成员变量有队列首尾元素的位置qfront、qback，容量arrayLength和储存队列元素的数组Queue。  (2)在扑克牌.h文件中定义一个采用默认构造函数的扑克牌类，成员变量有队列描述的牌堆，成员函数有实现卡片游戏的discard和输出output函数。Discard函数的实现思路是用户先输入一个数初始化牌堆，然后不断用队列的pop操作扔掉队列首元素（第一张牌）并将它输出，接着用pop和push操作将队列次元素（第二张牌）放入队列末尾（牌堆底），直到剩下最后一张牌。   1. 测试结果（测试输入，测试输出，结果分析）     如图，输入牌数10，然后依次输出被扔掉的牌，最后输出留下的一张牌。   1. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）   本实验较为简单，关键在于理解题目意思，所以基本没有遇到什么困难。   1. 附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）   linkedQueue.h  #pragma once  #include<iostream>  #include<cstdlib>  #include<algorithm>  template<class T>  struct chainNode  {  T element;  chainNode \*next=NULL;    chainNode(T ele,chainNode \*p)  {  element=ele;  next=p;  }  chainNode(T ele)  {  element=ele;  }  chainNode() { }  chainNode& operator=(chainNode &A)  {//结点赋值  element=A.element;  next=A.next;  return \*this;  }  };  template<class T>  class linkedQueue  {  public:  linkedQueue()  {//构造函数  queueFront=queueBack=NULL;  queueSize=0;  }  ~linkedQueue()  {//析构函数  chainNode<T>\* deleteNode = queueFront;  chainNode<T>\* stepNode;  while (deleteNode != NULL)  {  stepNode=deleteNode->next;  delete deleteNode;  stepNode = deleteNode;  }  }  bool empty() const  {  if(queueSize==0) return true;  else return false;  }  int size() const  {  return queueSize;  }  T& front()  {  if(queueSize!=0)  return queueFront->element;  else  { std::cerr<<"队列为空"<<std::endl; exit(1); }  }  T& back()  {  if(queueSize!=0)  return queueBack->element;  else  { std::cerr<<"队列为空"<<std::endl; exit(1); }  }  T pop()  {//从队首删除元素  if(queueSize==0)  {//队列为空  std::cerr<<"队列为空，禁止删除"<<std::endl;  exit(1);  }  else  {  chainNode<T> \*p=queueFront->next;  delete queueFront;  p=queueFront;  queueSize--;  }  }  void push(const T& ele)  {//从队尾添加元素  chainNode<T> \*p=new chainNode<T>(ele,NULL); //初始化    if(queueSize==0)  {//队列为空  queueBack=queueFront=p;  }  else{  queueBack->next=p;  queueBack=p;  }  queueSize++;  }  void output()  {  chainNode<T>\* curNode=queueFront;  for(int i=0;i<queueSize;i++)  {  std::cout<<curNode.element<<" ";  }  std::cout<<std::endl;  }  private:  chainNode<T>\* queueFront; // 首元素指针  chainNode<T>\* queueBack; // 尾元素指针  int queueSize; // 队列大小  };  扑克牌.h  #pragma once  #include<iostream>  #include"linkedQueue.h"  using namespace std;  template<class T>  class poker  {  public:  void discard(int num)  {//num是总牌数  for(int i=1;i<=num;i++)  {//对牌堆进行初始化  card\_pile.push(i);  }  while(card\_pile.size()>=2)  {  cout<<card\_pile.pop()<<" "; //扔掉第一张，并输出  T temp=card\_pile.pop();  card\_pile.push(temp); //把第二张插到最后  }  cout<<"剩下的牌是："; card\_pile.output();  }  void output()  {//输出牌堆中的元素  card\_pile.output();  }  private:  arrayQueue<T> card\_pile;    };  Main.cpp  #include<iostream>  #include"arrayQueue.h"  #include"扑克牌.h"  using namespace std;  int main()  {  poker<int> Poker; int num;  cout<<"请输入牌堆中牌的数量"<<endl;  cin>>num;  Poker.discard(num);      return 0;  } | | | |