**附件：**

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：202000130138 | 姓名：宋璎航 | | 班级： 20.3 |
| 实验题目：队列 | | | |
| 实验学时：2 | | 实验日期： 2021/11/11 | |
| 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。 | | | |
| 软件开发环境：  Vsc | | | |
| 1. 实验内容   **题目描述**：  首先创建队列类，采用数组描述；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，输出最后剩下的牌。  **输入输出格式**：  **输入**：  一个整数n，代表一开始卡片的总数。  **输出**：  最后一张卡片的值。   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）   简单模拟即可   1. 测试结果（测试输入，测试输出）  输入 100  **输出**  72   1. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）   测试结果正确   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）   #include "bits/stdc++.h"  template <class T>  class arrayQueue  {  T \*queue;  int queueFront, queueSize, arrayLength;  public:  arrayQueue()  {  queue = new T[10000];  arrayLength = 10000;  queueFront = 0;  queueSize = 0;  }  arrayQueue(int initialCapicity)  {  if (initialCapicity < 0)  exit(-1);  arrayLength = initialCapicity;  queue = new T[initialCapicity];  queueFront = 0;  queueSize = 0;  }  ~arrayQueue() { delete[] queue; }  bool empty() { return queueSize == 0; }  bool full() { return queueSize == arrayLength; }  int size() { return queueSize; }  void push(const T &theElement);  void pop();  void output() const;  T &front() const { return queue[queueFront]; }  T &back() const { return queue[(queueFront + queueSize) % arrayLength]; }  };  template <class T>  void arrayQueue<T>::push(const T &theElement)  {  if (full())  {  T \*temp = new T[2 \* arrayLength];  int cnt = 0;  while (cnt < queueSize)  {  temp[cnt] = queue[(queueFront + cnt) % arrayLength];  cnt++;  }  delete[] queue;  queue = temp;  queueFront = 0;  arrayLength \*= 2;  }  queue[(queueFront + queueSize) % arrayLength] = theElement;  queueSize++;  }  template <class T>  void arrayQueue<T>::pop()  {  if (empty())  exit(-1);  queue[queueFront].~T();  queueFront += 1;  queueFront %= arrayLength;  queueSize -= 1;  }  template <class T>  void arrayQueue<T>::output() const  {  for (size\_t i = 0; i < queueSize; ++i)  std::cout << queue[(queueFront + i) % arrayLength] << std::endl;  }  int main()  {  arrayQueue<int> a;  int n;  std::cin >> n;  for (size\_t i = 1; i <= n; ++i)  a.push(i);  while (a.size() >= 2)  {  a.pop();  int temp = a.front();  a.pop();  a.push(temp);  }  a.output();  } | | | |