项目说明文档

数据结构课程设计

——修理牧场

作 者 姓 名： 杨晶

学 号： 1854025

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目 录

[1 分析](#_Toc495668153)

[1.1 问题简介](#_Toc495668154)

[1.2 思路分析](#_Toc495668155)

[2 设计](#_Toc495668156)

[2.1 数据结构设计](#_Toc495668157)

[2.2 求解函数设计](#_Toc495668158)

[2.3 主函数设计](#_Toc495668159)

[3 实现](#_Toc495668161)

[3.1 优先级队列的实现](#_Toc495668162)

[3.1.1 插入函数实现](#_Toc495668163)

[3.1.2 删除函数实现](#_Toc495668164)

[3.2 求最小花费函数的实现](#_Toc495668166)

[3.3 主函数的实现](#_Toc495668170)

[4 测试](#_Toc495668186)

[4.1 功能测试](#_Toc495668187)

[4.1.1测试](#_Toc495668188)一

[4.1.2测试](#_Toc495668189)二

[4.1.3测试](#_Toc495668190)三

[4.2 边界测试](#_Toc495668193)

[4.2.1 初始化数据为0](#_Toc495668194)

[4.2.2 初始化一块木头](#_Toc495668195)

# 1 分析

## 1.1 问题简介

农夫要修理牧场的一段栅栏，他测量了栅栏，发现需要N块木头，每块木头长度为整数Li个长度单位，于是他购买了一个很长的，能锯成N块的木头，即该木头的长度是Li的总和。

但是农夫自己没有锯子，请人锯木的酬金跟这段木头的长度成正比。为简单起见，不妨就设酬金等于所锯木头的长度。例如，要将长度为20的木头锯成长度为8，7和5的三段，第一次锯木头将木头锯成12和8，花费20；第二次锯木头将长度为12的木头锯成7和5花费12，总花费32元。如果第一次将木头锯成15和5，则第二次将木头锯成7和8，那么总的花费是35（大于32）。

项目功能要求：

（1） 输入格式：输入第一行给出正整数N（N<10000），表示要将木头锯成N块。第二行给出N个正整数，表示每块木头的长度。

（2） 输出格式：输出一个整数，即将木头锯成N块的最小花费。

## 1.2 思路分析

给了要将木头锯成N块,与每块木头的长度。最后需要输出最小的花费。

刚开始时，我的思路是因为是从一块木头开始分割的，且每次都将木块分割为两块，无非是大小不同而已。通过反向思考,如果每次对半分的时候,最后结果最小，那么,在只有碎片的时候,每一次都拿出最小的两个进行拼凑，就是对半分的逆过程了， 所以只需要有一个每次都能取出最小元素的优先队列即可，所有就只要构建一个从小到大的优先队列就能解决问题了。

# 2 设计

## 2.1 数据结构设计

根据上述分析，此题我们采用最小优先级队列实现。最小优先级队列是指，如果队列中的某个元素最小，那么应该先对它进行操作。

为了实现这个操作，可以构建一个栈，在向栈数组中插入元素时保证数组中元素顺序从小到大即可。

## 2.2 求解函数设计

求解函数中，利用数组中已经按从小到大有序排列的性质，将第一个和第二个元素相加，将两者之和插入队列的同时在原队列中删除这两个元素。同时将两者的和加入最终花费上，对所有数据处理一遍后即可得到最终结果。

## 2.3 主函数设计

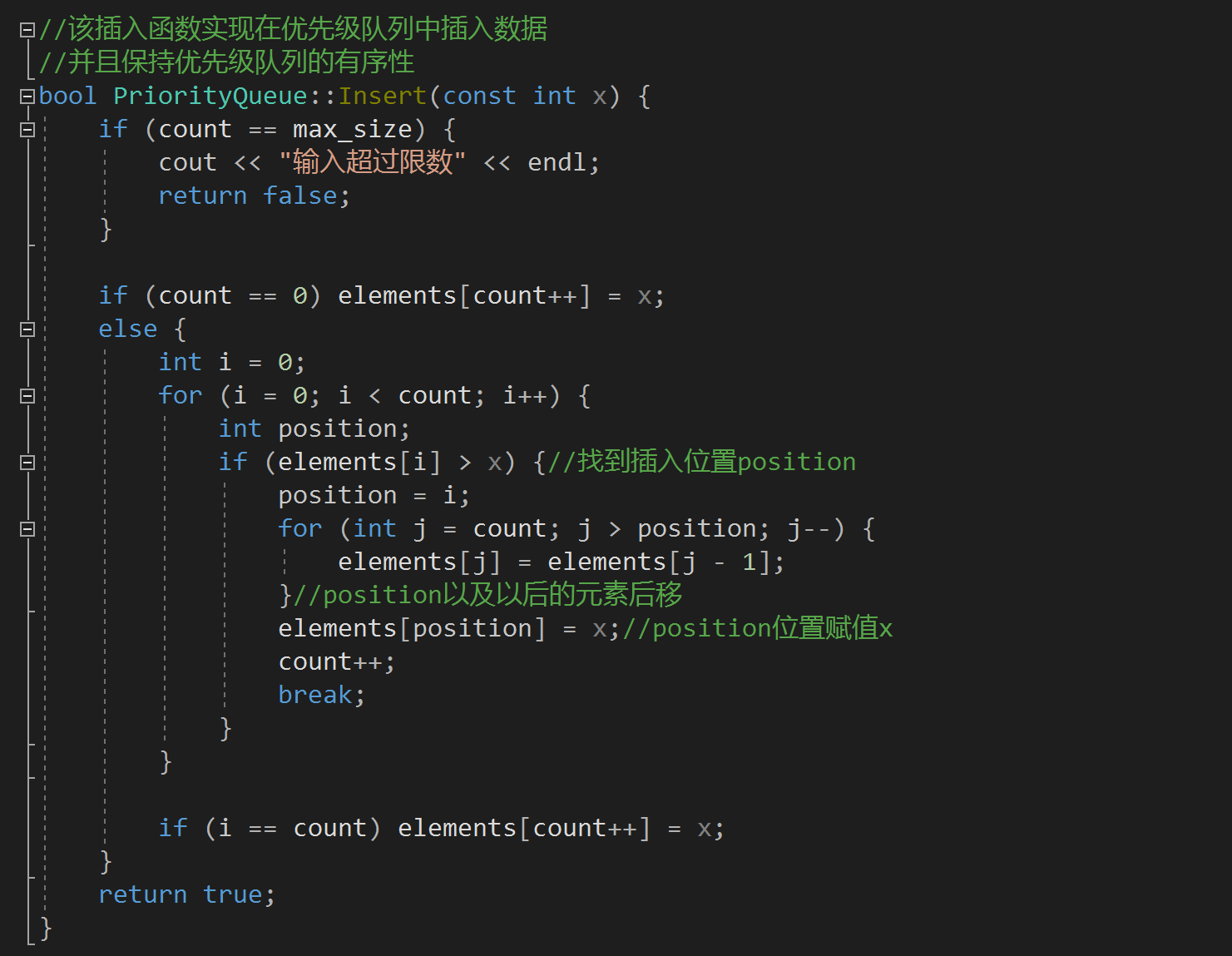
主函数主要负责队列初始化，数据输入。然后调用求解函数输出结果。

# 3 实现

## 3.1 优先级队列的实现

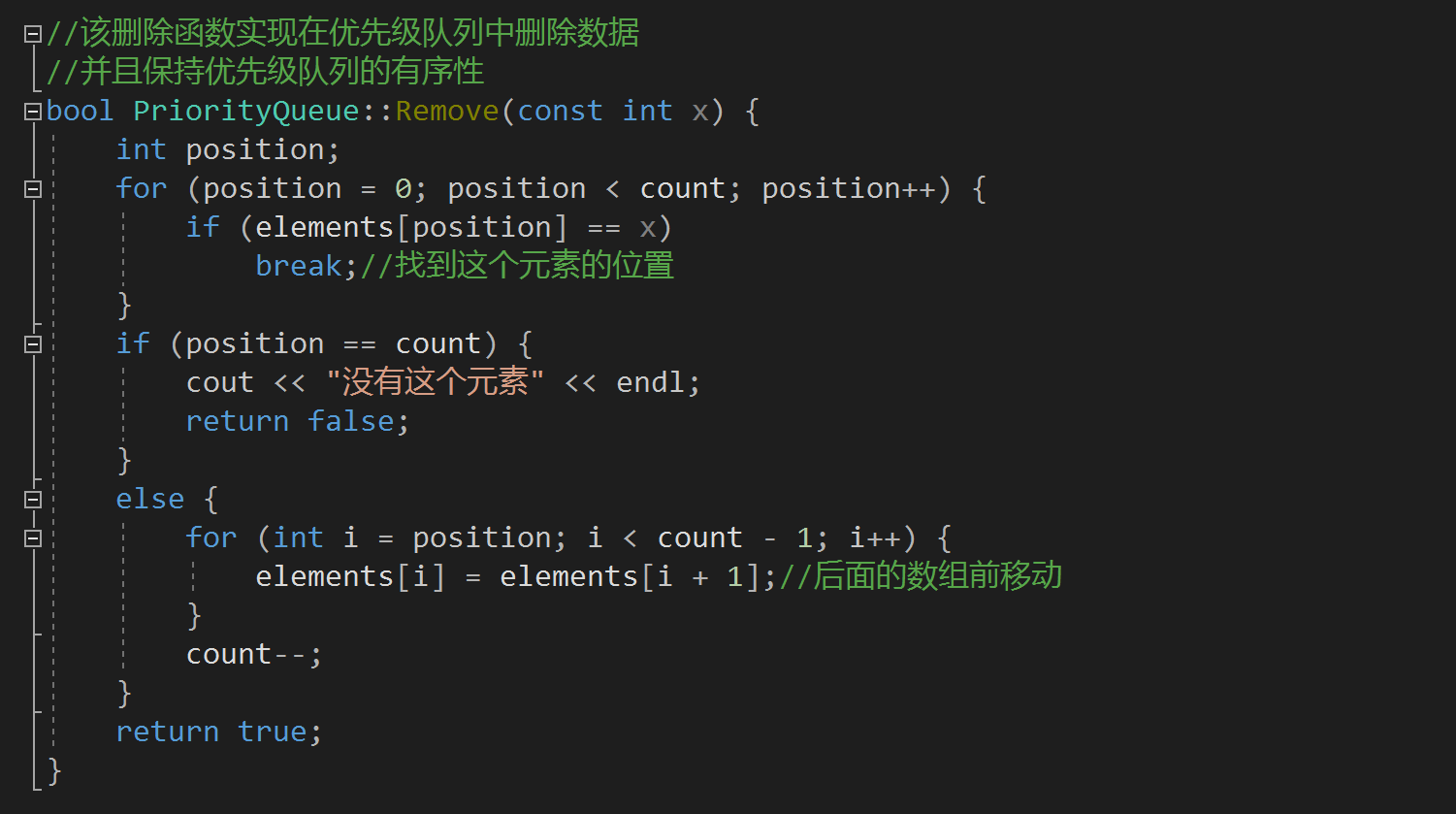
### 3.1.1 插入函数实现

插入函数首先判断数据栈是否已满，如果已满就输出报错信息。接着判断数据栈中是否有数据，如果原先没有数据就直接插入到数据栈中，如果以前有数据，就先按大小顺序找到应该插入的位置，记为position。然后将position后面的数组元素依次往后移动，将目标位置空出后进行插入，同时count自加。



### 3.1.2 删除函数实现

删除函数首先要找到要删除元素的目标位置，记为position，如果没有找到，则输出报错信息。找到元素后，从这个元素开始，将元素之后的那个值赋值给这个元素，同时count自减。

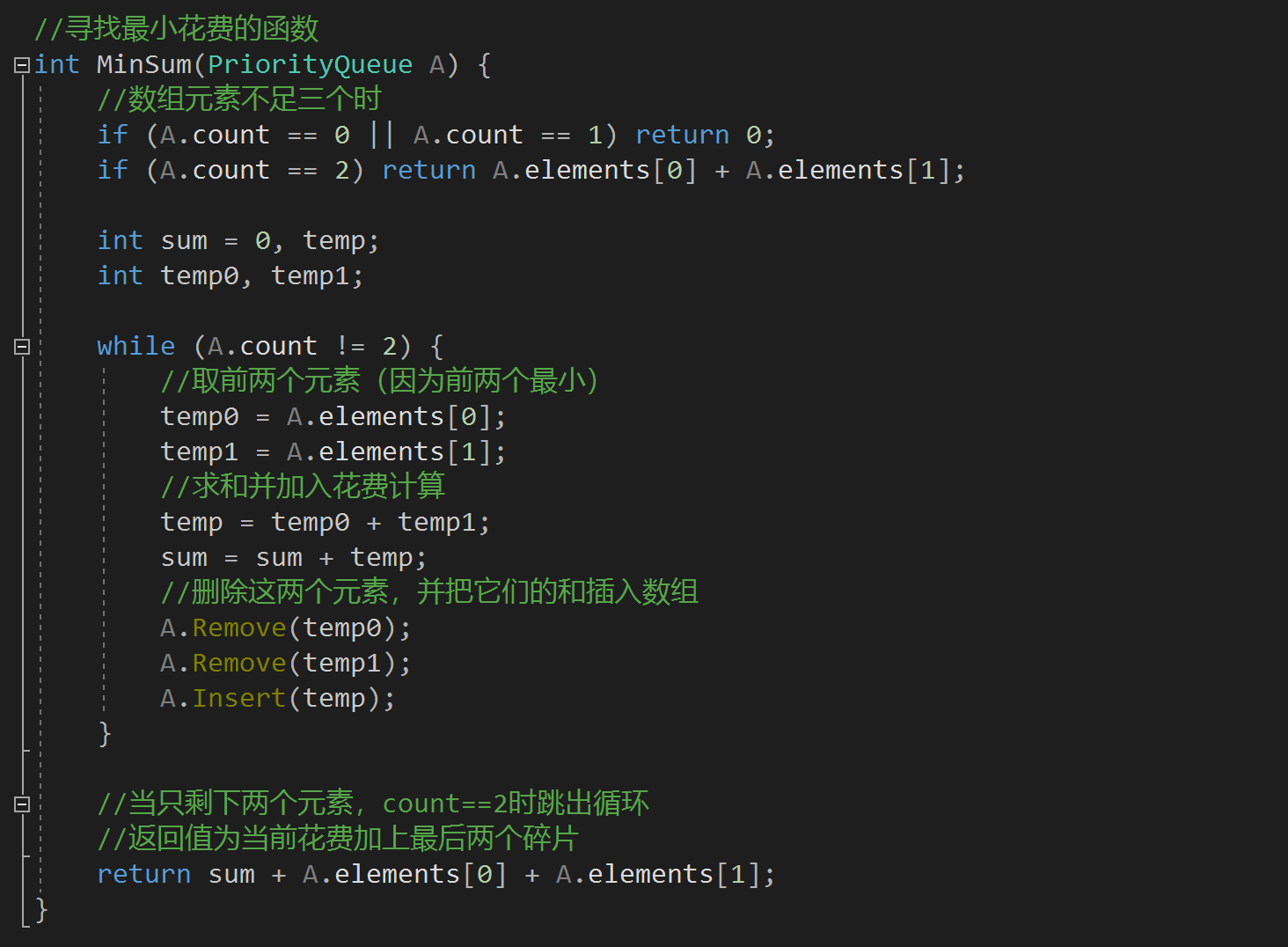


## 3.2 求最小花费函数的实现

求解函数中，首先判断当前数据栈有多少个元素，如果只有0个或者1个元素，则直接返回0或者该元素。如果有两个元素，则返回两个元素的和。

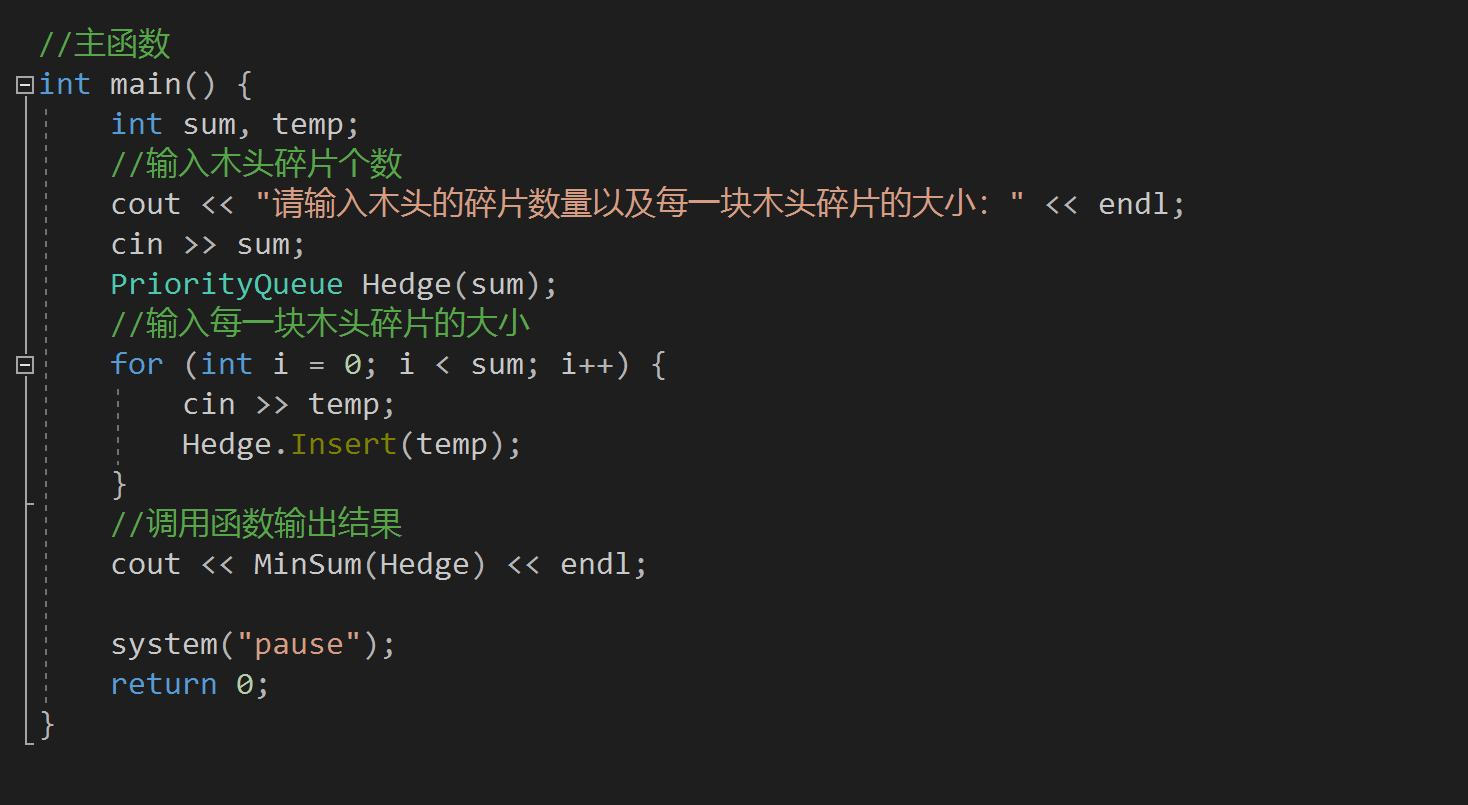
如果数据栈中有很多元素，则按照我们逆向思考的思路，每次拿出数据栈中最小的两个相加，相加的和即为此次切割需要的花费。接着把这两块木块从数据栈中弹出，将刚刚两者之和push进栈即可。

设定循环终止的条件为数据栈中剩两个元素，当只有两个元素时，就直接拿最后相加之和与最后两个元素相加即可。



## 3.3 主函数的实现

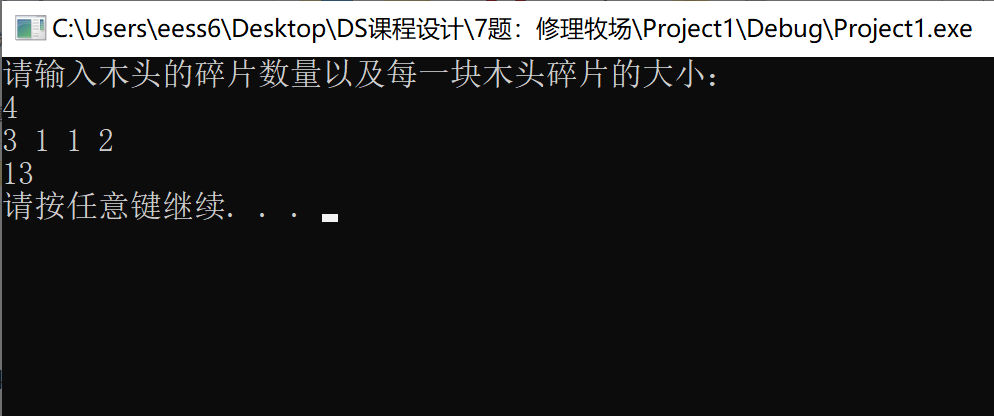
主函数主要实现木块数量的输入，以及每一块木块碎片的大小输入



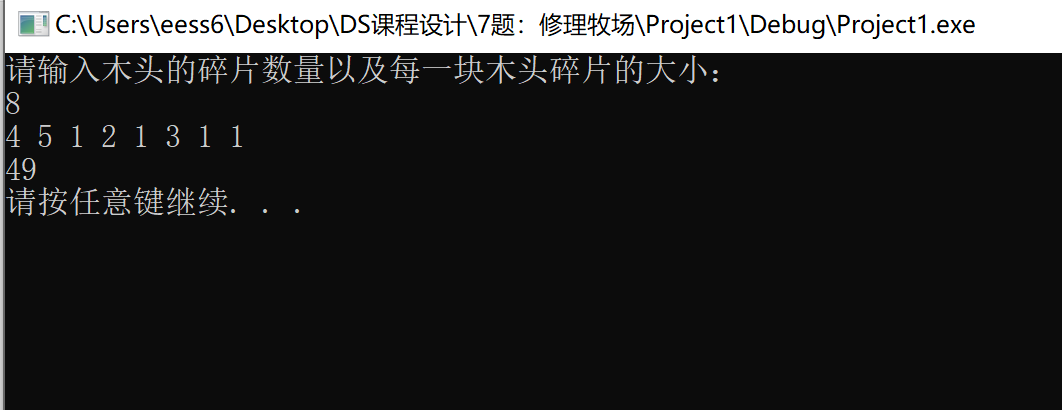
# 4 测试

## 4.1 功能测试

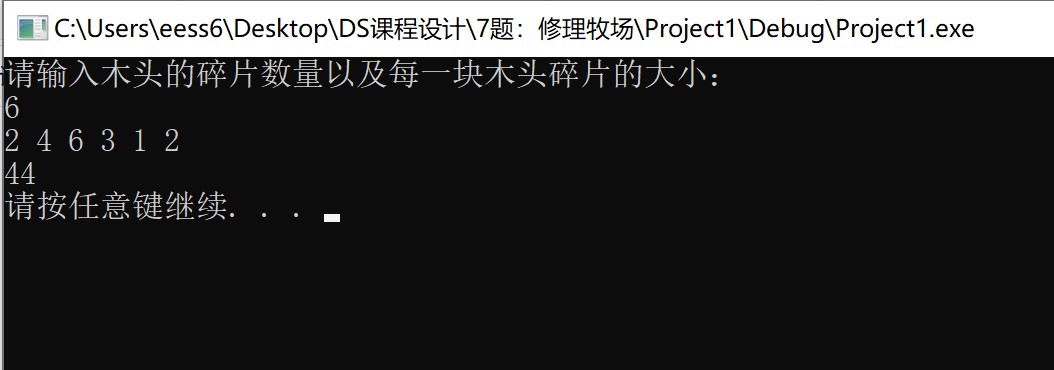
### 4.1.1 测试一



### 4.1.2 测试二

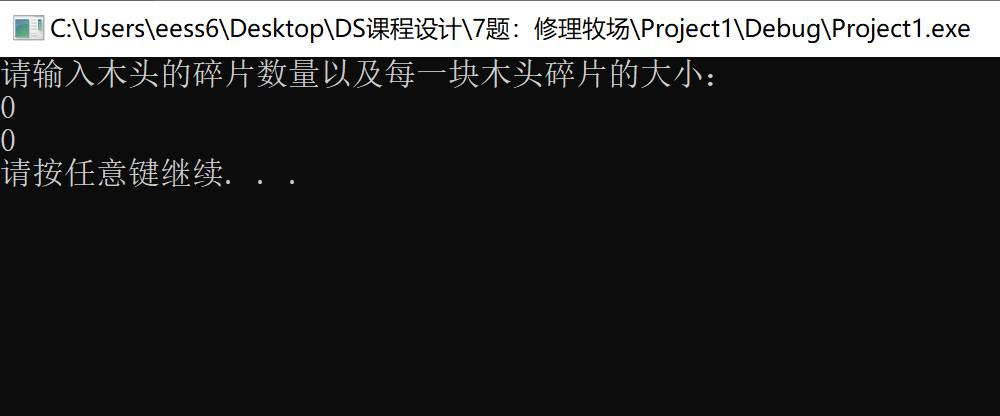


### 4.1.3 测试三



## 4.2 边界测试

### 4.2.1 初始化数据为0



### 4.2.2 初始化一块木头

