项目说明文档

数据结构课程设计

——修理牧场

作 者 姓 名： 李子涵

学 号： 1851892

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 3](#_Toc32415)

[1.1 背景分析 3](#_Toc437)

[1.2 功能分析 3](#_Toc740)

[2 设计 4](#_Toc881)

[2.1 数据结构设计 4](#_Toc19246)

[2.2 类结构设计 4](#_Toc22206)

[2.3 成员与操作设计 4](#_Toc3817)

[2.4 系统设计 4](#_Toc16235)

[3 实现 5](#_Toc10075)

[3.1 插入功能的实现 5](#_Toc9829)

[3.1.1 插入功能流程图 5](#_Toc13798)

[3.1.2 插入功能核心代码 5](#_Toc3571)

[3.2 清除功能的实现 7](#_Toc11471)

[3.2.1 清除功能流程图 7](#_Toc27425)

[3.2.2 清除功能核心代码 7](#_Toc28584)

[3.3 计算功能的实现 8](#_Toc13069)

[3.3.1 计算功能流程图 8](#_Toc1957)

[3.3.2 计算功能核心代码 8](#_Toc14458)

[3.4 输入功能的实现 10](#_Toc14671)

[3.4.1 输入功能流程图 10](#_Toc24384)

[3.4.2 输入功能核心代码 10](#_Toc3282)

[3.6 总体程序的实现 11](#_Toc7132)

[3.6.1 总体程序流程图 11](#_Toc1449)

[3.6.2 总体系统核心代码 12](#_Toc10484)

[3.6.3 总体程序截屏示例 12](#_Toc27825)

[4 测试 13](#_Toc27020)

[4.1 功能测试 13](#_Toc26615)

[4.1.1 测试1 13](#_Toc26165)

[4.1.2 测试2 13](#_Toc21428)

[4.2 边界测试 14](#_Toc11534)

[4.2.1 总数N为1 14](#_Toc24885)

[4.3 出错测试 14](#_Toc28234)

[4.3.1 总数为负数 14](#_Toc1183)

[4.3.2 总数N大于10000 15](#_Toc20256)

[4.3.3 长度有负数 15](#_Toc1257)

# 1 分析

## 1.1 背景分析

修理牧场这一题目背景很具有现实意义。在已知原材料木头总长度、所需木头段数和每段木头的情况下，采用每锯断一次木头就收取相应总长度单位额的手工费，如何锯木头才能使得总开销最小。完成这一计算，对于某一给出的特定条件来说并不复杂，但要大量重复地计算不同条件下问题的结果就比较麻烦。

因此，可以设计一个计算机程序来解决这一问题，利用可重复利用的算法快速灵活地计算出不同条件下的最优解，可以大大提高解决此类问题的效率。

## 1.2 功能分析

本程序功能简单，只需用户输入所需的总数和每段长度，再完成计算和输出即可。

因此，程序应该具有输入、输出、计算的功能。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，本程序需要采用贪心算法，对一系列长度排序后从小到大两两相加、再将和按顺序插入已排好的数据结构。最适合在插入同时排序的数据结构是堆，但考虑到使用的算法需要不停按顺序向后访问，使用链表结构更加方便。

## 2.2 类结构设计

本程序使用struct描述链表结点类（Node），使得链表类（list）可以方便地访问链表结点，又确保了储存在class private中的头结点和尾指针不会丢失。

## 2.3 成员与操作设计

**链表结点struct（Node）**

int value;//长度数值

Node\* next;//指向下一结点

**链表类（list）**

**私有成员：**

Node\* head;//头指针

Node\* tail;//尾指针

**公有操作：**

list();//构造函数

~list();//析构函数

bool insert(int&);//按从小到大顺序插入新结点

bool is\_empty() { return head == tail; }//判断链表是否为空

bool sum\_cost();//计算并输入话费

bool clear\_list();//清空链表

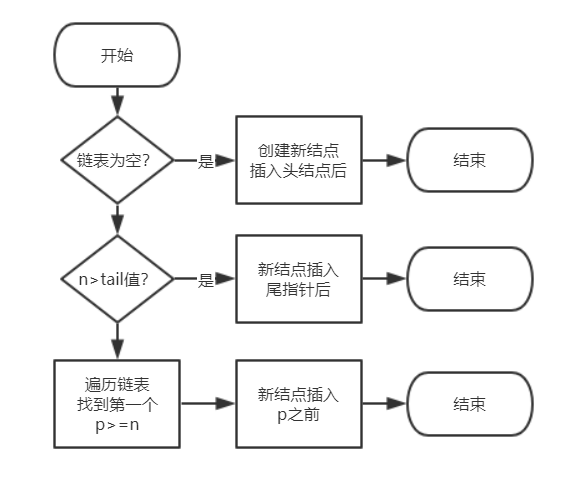
## 2.4 系统设计

程序首先完成对链表wood的创建和对屏幕的初始化，输入总数N，然后调用input函数实现长度数据的逐一输入，最后计算并输出结果。

# 3 实现

## 3.1 插入功能的实现

### 3.1.1 插入功能流程图



### 3.1.2 插入功能核心代码

Node\* h = new Node{ n,NULL };

if (is\_empty())//若链表为空,直接将新结点接在头指针后

{

tail->next = h;

tail = h;

return 1;

}

Node\* p = head->next, \* q = head;

if (tail->value <= n)//若尾指针数值不比新结点大,直接插在尾指针后

{

tail->next = h;

tail = h;

return 1;

}

while (p->value < n)//从头遍历结点,找到恰好大于等于新结点的结点

{

p = p->next;

q = q->next;

}

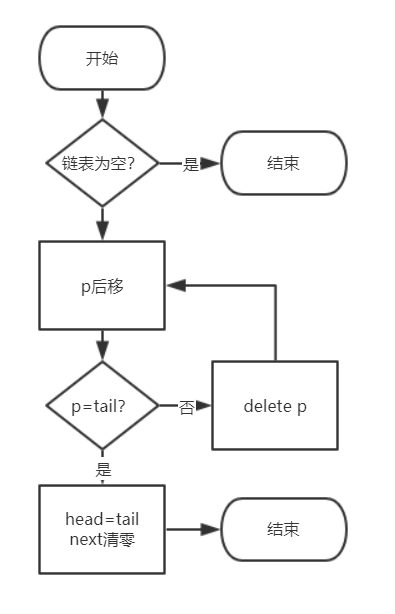
h->next = p;//插入新结点

q ->next = h;

return 1;

## 3.2 清除功能的实现

### 3.2.1 清除功能流程图



### 3.2.2 清除功能核心代码

if (is\_empty())

return 0;

Node\* p = head->next;

while (head != tail)

{

delete head;

head = p;

p = p->next;

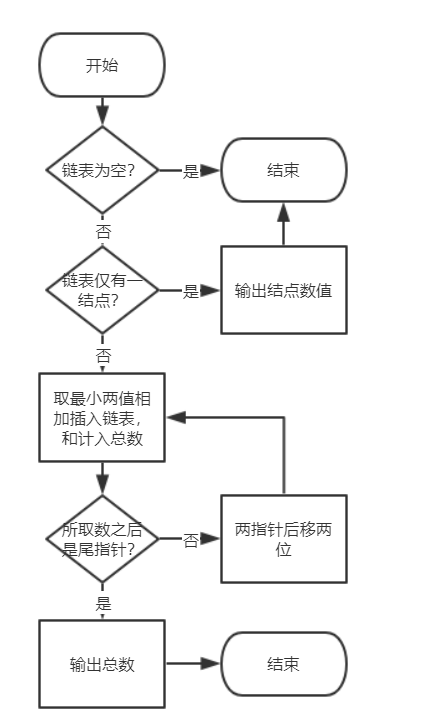
}

head->next = NULL;

return 1;

## 3.3 计算功能的实现

### 3.3.1 计算功能流程图



### 3.3.2 计算功能核心代码

if (is\_empty())//链表为空,返回0

return 0;

if (head->next == tail)//若仅有一个结点,结点数值即为所求,直接输出

{

cout << tail->value << endl;

return 1;

}

int cost = 0;

Node\* left = head->next, \* right = left->next;//成对指针

while (right != tail)

{

int s = (right->value) + (left->value);

insert(s);//最小两长度相加,插入链表

cost += s;//两长度计算入总花销

left = right->next;//两指针后移两位

right = left->next;

}

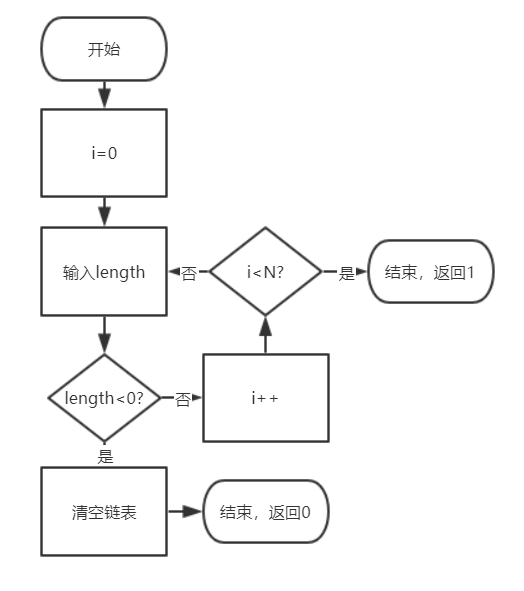
cost = cost + left->value + right->value;//最后两节长度计算入总花销

cout << cost << endl;//输出花销

return 1;

## 3.4 输入功能的实现

### 3.4.1 输入功能流程图



### 3.4.2 输入功能核心代码

int length;//每段长度

for (int i = 0; i < rN; i++)//循环输入N次

{

cin >> length;//输入每段长度

if (length <= 0)//若输入不是正整数,报错,退出函数

{

rwood.clear\_list();//调用函数清除list结点

fflush(stdin);//清空缓存区

return 0;//返回0值

}

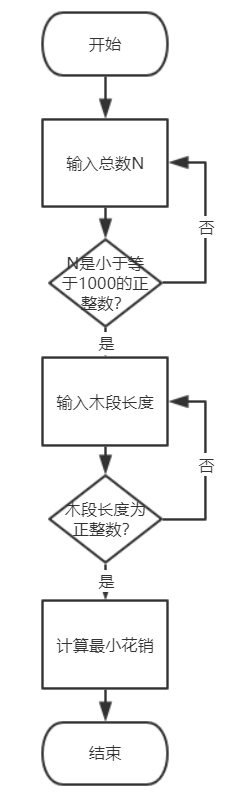
rwood.insert(length);//插入对应长度结点

}

return 1;

## 3.6 总体程序的实现

### 3.6.1 总体程序流程图



### 3.6.2 总体系统核心代码

list wood;//存储木头段长度的链表

cout << "输入所需木头段数：" << endl;

int N;//木头段总数N

cin >> N;//输入N

while (N < 1 || N > 10000)//若输入不是小于等于10000的正整数,重新输入

{

cout << "请输入小于10000的正整数：" << endl;

cin >> N;

}

cout << "输入各段木头长度：" << endl;

while (!input(N, wood))//输入各段长度,若输入为负数,重新输入

{

cout << "长度不能为负数，请重新输入：" << endl;

}

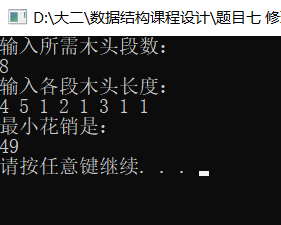
cout << "最小花销是：" << endl;

wood.sum\_cost();//计算并输出最小花销

system("pause");

return 1;

### 3.6.3 总体程序截屏示例



# 4 测试

## 4.1 功能测试

### 4.1.1 测试1

**测试用例**：

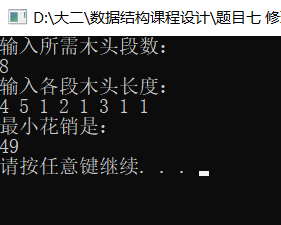
8

4 5 1 2 1 3 1 1

**预期结果**：

49

**实验结果**



### 4.1.2 测试2

**测试用例：**

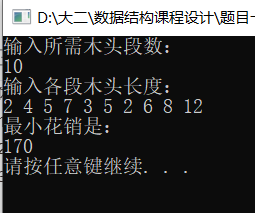
**10**

**2 4 5 7 3 5 2 6 8 12**

**预期结果：**

**170**

**实验结果：**



## 4.2 边界测试

### 4.2.1 总数N为1

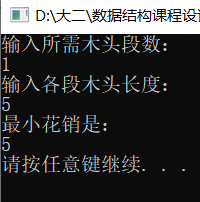
**测试用例：**

**1**

**5**

**预期结果：**程序运行正常不崩溃。

**实验结果：**



## 4.3 出错测试

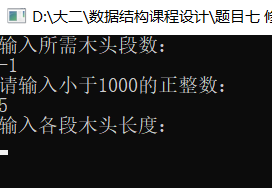
### 4.3.1 总数为负数

**测试用例：**

**-1**

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**



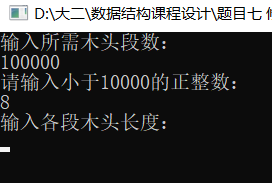
### 4.3.2 总数N大于10000

**测试用例：**

**100000**

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**



### 4.3.3 长度有负数

**测试用例：**

**5**

**1 2 4 -2 5**

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

