项目说明文档

数据结构课程设计

——两个有序链表的交集

作 者 姓 名： 曹峰源

学 号： 1951328

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](#_Toc57651754)

[2. 设计 3](#_Toc57651755)

[2.1 类结构设计 3](#_Toc57651756)

[2.2 类成员 3](#_Toc57651757)

[3. 功能实现 4](#_Toc57651758)

[3.1 构造链表功能实现 4](#_Toc57651759)

[3.2 寻找交集功能的实现 5](#_Toc57651760)

[3.3 主函数功能 7](#_Toc57651761)

[4. 功能测试 7](#_Toc57651762)

[4.1 两链表均不为空测试 7](#_Toc57651763)

[4.2 有链表为空测试 8](#_Toc57651764)

[5. 出错测试 8](#_Toc57651765)

1. 分析

第二题两个有序链表的交集较为简单，满足题目要求，首先要实现一个链表使得数据能够输入进去，然后就是能够进行题目要求的求出两个链表的交集。最后我们希望能够对初始链表进行一些判断，判断它是否满足升序这一条件。

1. 设计

## 2.1 类结构设计

题目已经要求使用链表，对于链表，我采用的是用两个类，一个node类，一个linklist类来实现代码功能。其中使linklist类成为node类的友元从而能访问node类的数据

## 2.2 类成员

链表结点node类：

私有成员：

T number=0;//链表中的元素 T是模板

node<T>\* next = NULL;//指向下一个结点的指针

链表linklist类：

私有成员：

node<T>\* first;//头指针

node<T>\* current;//用于移动的指针

公有函数：

linklist<T>(){};//构造函数

~linklist<T>(){};//析构函数

node<T>\* getfirst();//用于返回头指针

void insert(T a);//用于在链表最后插入元素

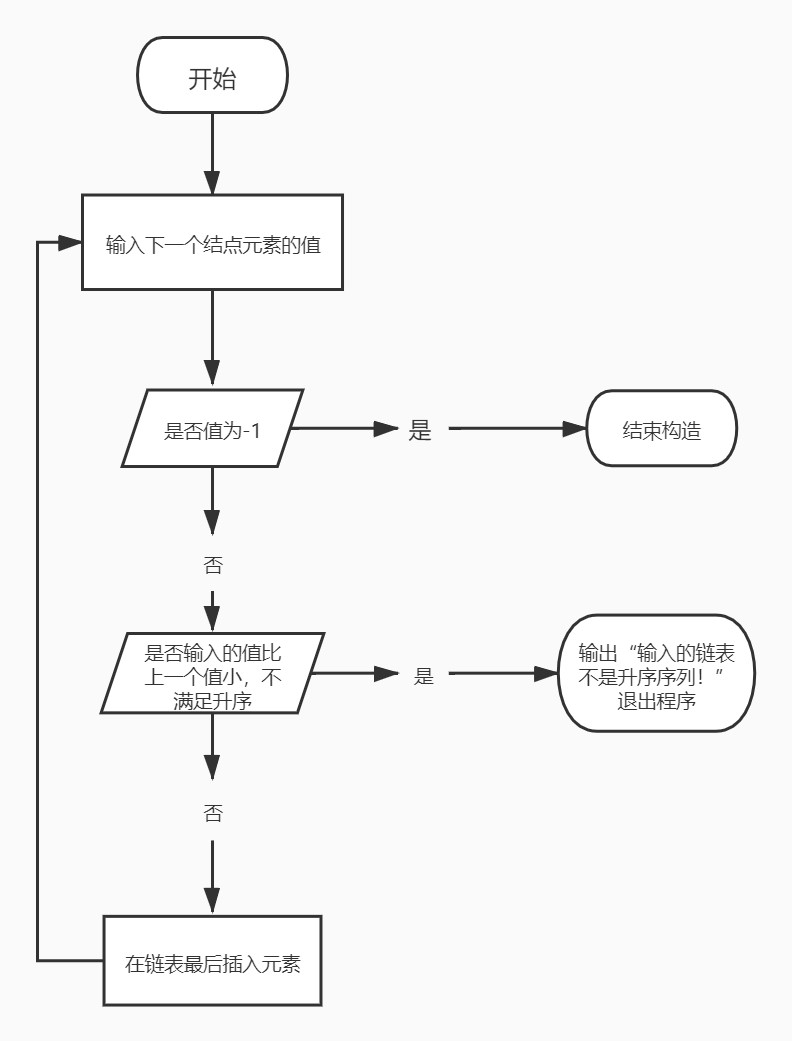
void create();//构造链表

void display();//打印链表元素

void combination(node<T>\* secondfirst, node<T>\* thirdfirst);//用于实现题目要求，将两个链表的头指针作为参数引用，将该链表与secondfirst指针的链表的公共元素放到thirdfirst指针的链表中

1. 功能实现
   1. 构造链表功能实现

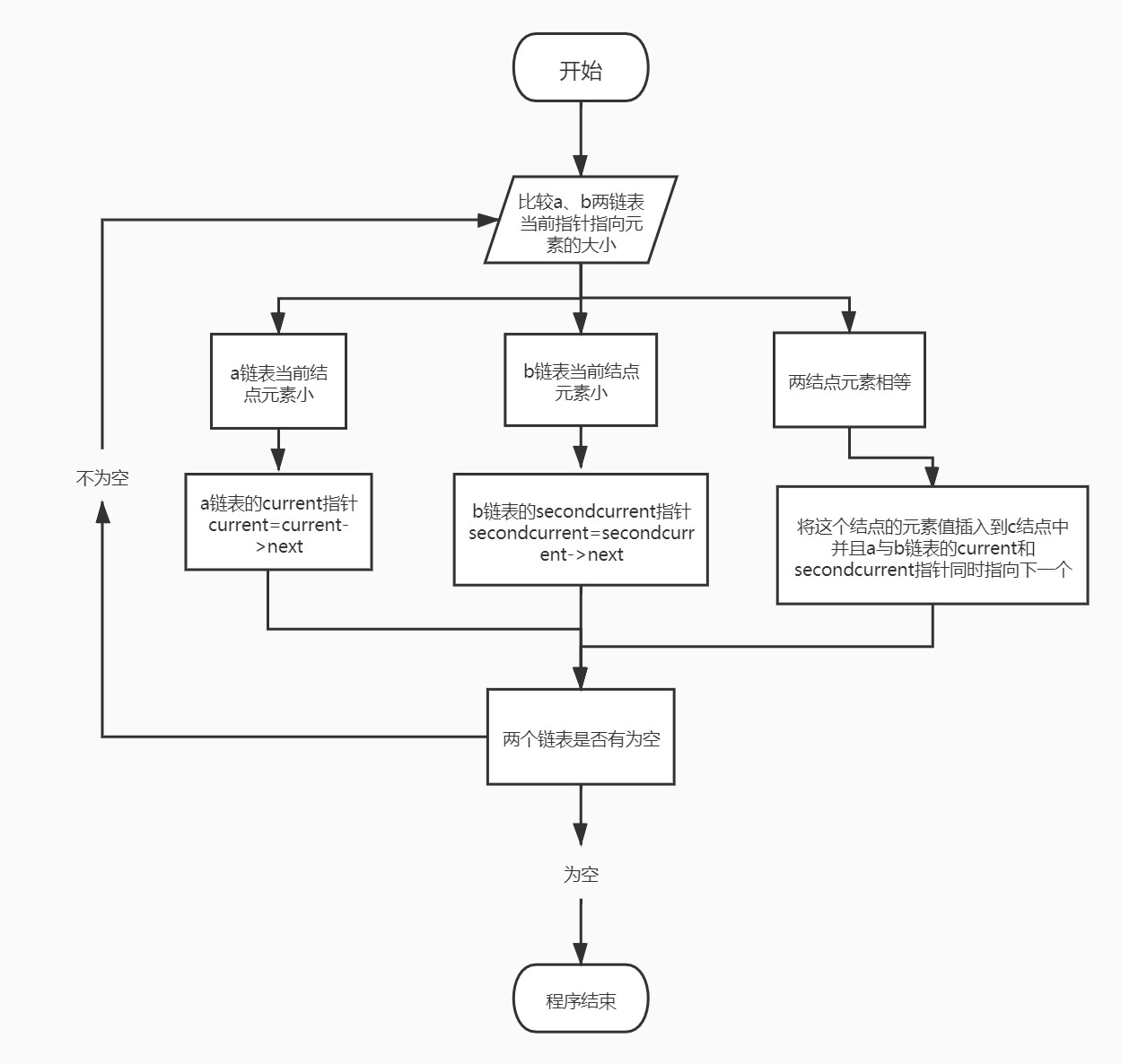
流程图：



插入功能代码：

1. **void** linklist<T>::create()
2. {
3. T n;
4. //输入n，当n为-1时，跳出循环，停止输入
5. **while** (cin >> n)
6. {
7. //先将current指针指向头结点
8. current = first;
9. //遍历链表，让current指针指向链表最后一个节点
10. **while** (current->next != NULL)
11. {
12. current = current->next;
13. }
14. **if** (n != -1&¤t->number<=n)
15. {
16. insert(n);
17. }
18. **else** **if** (n!=-1&¤t->number>n)
19. {
20. cout << "输入的链表不是升序序列！";
21. exit(0);
22. }
23. **else** **break**;
24. }
25. }
    1. 寻找交集功能的实现

流程图：



寻找交集代码：

1. **void** linklist<T>::combination(node<T>\* secondfirst, node<T>\* thirdfirst)
2. {
3. current = first->next;
4. //创建一个secondcurrent指针指向第二个链表头指针的下一个节点
5. node<T>\* secondcurrent = secondfirst->next;
6. //创建一个thirdcurrent指针指向第三个链表的头结点
7. node<T>\* thirdcurrent;
8. thirdcurrent = thirdfirst;
9. //count用于计数，当count为0时说明没有一个节点数据是相同的，此时输出NULL
10. **int** count = 0;
11. **while** (current != NULL && secondcurrent != NULL)
12. {
13. //当一个节点的元素比另一个大时，小的那个节点指针右移
14. **if** (current->number > secondcurrent->number)
15. {
16. secondcurrent = secondcurrent->next;
17. }
18. **else** **if** (current->number < secondcurrent->number)
19. {
20. current = current->next;
21. }
22. //出现相等的情况时，将这个节点的元素保存在新的链表中
23. **else**
24. {
25. node<T>\* newnode = **new** node<T>;
26. newnode->number = current->number;
27. newnode->next = thirdcurrent->next;
28. thirdcurrent->next = newnode;
29. thirdcurrent = thirdcurrent->next;
30. current = current->next;
31. secondcurrent = secondcurrent->next;
32. count++;
33. }
34. }
35. **if** (count == 0) cout << "NULL";
36. }
    1. 主函数功能

定义三个链表，输入前两个链表的值，将交集放到第三个链表中

代码：

1. **int** main()
2. {
3. linklist<**double**> list1, list2, list3;//主函数中暂时将模板实例化为double类型
4. list1.create();
5. list2.create();
6. node<**double**>\* p = list2.getfirst();
7. node<**double**>\* q = list3.getfirst();
8. list1.combination(p, q);
9. list3.display();
10. **return** 0;
11. }
12. 功能测试

用double类型进行测试

4.1 两链表均不为空测试

测试用例：0.1 0.2 1.23 2.4 -1

0.2 0.4 1.23 -1

结果：



4.2 有链表为空测试

测试用例：0.1 0.2 1.23 2.4 -1

-1

及-1

0.2 0.4 1.23 -1

结果：





1. 出错测试

输入的不是升序序列

预期结果：程序不崩溃，输出提示错误信息后退出程序

用double类型进行测试

测试用例：0.1 0.2 1.23 0.82 -1

结果：

