项目说明文档

数据结构课程设计

——两个有序链表序列的交集

作 者 姓 名： 赵子昱

学 号： 1951459

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 1](#_Toc59383651)

[1.1 功能要求 1](#_Toc59383652)

[1.2 功能分析 1](#_Toc59383653)

[2 设计与实现 1](#_Toc59383654)

[2.1 **Intersection(…)**函数的实现 1](#_Toc59383655)

[2.1.1 **Intersection(…)**函数流程图 2](#_Toc59383656)

[2.1.2 **Intersection(…)**函数代码 2](#_Toc59383657)

[2.2 **test()**函数实现 3](#_Toc59383658)

[2.2.1 **test()**函数代码 3](#_Toc59383659)

[2.3 主函数的实现 3](#_Toc59383660)

[2.3.1 主函数代码 3](#_Toc59383661)

[3 测试 4](#_Toc59383662)

[3.1 功能测试 4](#_Toc59383663)

[3.1.1 测试1 4](#_Toc59383664)

[3.1.2 测试2 4](#_Toc59383665)

[3.1.3 测试3 4](#_Toc59383666)

[3.1.4 测试4 5](#_Toc59383667)

[3.2 边界测试 5](#_Toc59383668)

[3.2.1 其中一个链表为空 5](#_Toc59383669)

[3.2.2 两个链表为空 6](#_Toc59383670)

# 1 分析

## 1.1 功能要求

已知两个非降序链表序列S1和S2，设计函数构造出S1和S2的交集新链表S3。

1. 输入说明：输入分2行，分别在每行给出由若干个正整数构成的非降序序列，用-1表示序列的结尾（-1不属于这个序列）。数字用空格间隔。
2. 输出说明：在一行中输出两个输入序列的交集序列，数字间用空格分开，结尾不能有多余空格；若新链表为空，输出NULL。
3. 测试用例：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 输入 | 输出 | 说明 |
| 1 | 1 2 5 -1  2 4 5 8 10 -1 | 2 5 | 一般情况 |
| 2 | 1 3 5 -1  2 4 6 8 10 -1 | NULL | 交集为空的情况 |
| 3 | 1 2 3 4 5 -1  1 2 3 4 5 -1 | 1 2 3 4 5 | 完全相交的情况 |
| 4 | 3 5 7 -1  2 3 4 5 6 7 8 -1 | 3 5 7 | 其中一个序列完全属于交集的情况 |
| 5 | -1  10 100 1000 -1 | NULL | 其中一个序列为空的情况 |

## 1.2 功能分析

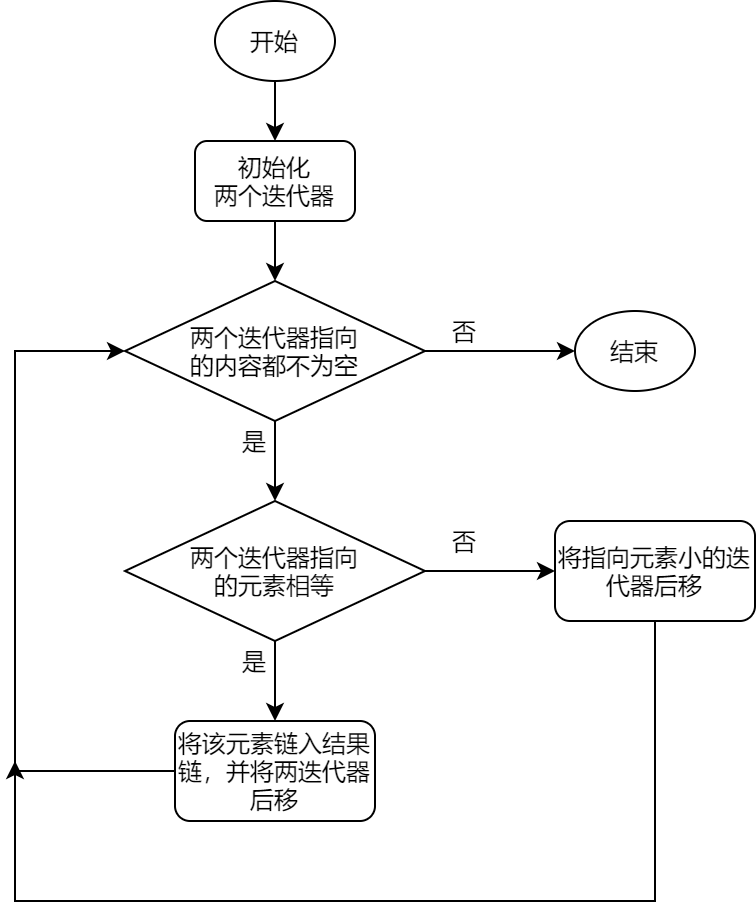
实现有序链表求交集，首先需要两个链表来分别储存原始信息，为了不破坏初始链表，新开一个**result**链表来储存结果。利用两个**list1**，**list2**的迭代器，来实现链表的遍历。从链表头开始逐渐对比，并且把相同的元素链入**result**，直到走到其中一个链表的尽头。

# 2 设计与实现

## 2.1 **Intersection(…)**函数的实现

* **Intersection(…)**函数为整个项目的主要函数
* **list1，list2**为两个原始链表，**result**为结果链表
* 用两个迭代器来实现链表的遍历
* 将指向元素小的迭代器后移一位，若相等则将该元素链入结果链
* 循环，直到遍历完某一链表

### 2.1.1 **Intersection(…)**函数流程图



### 2.1.2 **Intersection(…)**函数代码

1. **void** Intersection(**const** List<**float**> &list1, **const** List<**float**> &list2,List<**float**> &result) {
2. ListIterator<**float**> iter1(list1), iter2(list2);
3. **float** \*num1 = iter1.Next(), \*num2 = iter2.Next();
4. **while** (num1 && num2) {
5. **if** (\*num1 < \*num2) {
6. num1 = iter1.Next();
7. }
8. **else** **if** (\*num1 > \*num2) {
9. num2 = iter2.Next();
10. }
11. **else** {
12. result.Insert(\*num1, -1);
13. num1 = iter1.Next();
14. num2 = iter2.Next();
15. }
16. }
17. }

## 2.2 **test()**函数实现

* test()函数用于外部调用，包含了输入，求交集，展示结果

### 2.2.1 **test()**函数代码

1. **void** test(){
2. List<**float**> list1, list2, result;
3. **int** tempInt;
4. **while** (cin >> tempInt, tempInt != -1) {
5. list1.Insert(tempInt, -1);
6. }
7. **while** (cin >> tempInt, tempInt != -1) {
8. list2.Insert(tempInt, -1);
9. }
10. Intersection(list1, list2, result);
11. result.Show();
12. }

## 2.3 主函数的实现

### 2.3.1 主函数代码

1. **int** main() {
2. test();
3. cout << endl;
4. system("pause");
5. **return** 0;
6. }

# 3 测试

## 3.1 功能测试

### 3.1.1 测试1

**测试用例**：

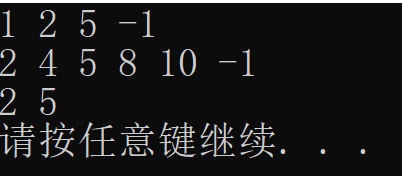
1 2 5 -1

2 4 5 8 10 -1

**预期结果**：

2 5

**实验结果**



### 3.1.2 测试2

**测试用例：**

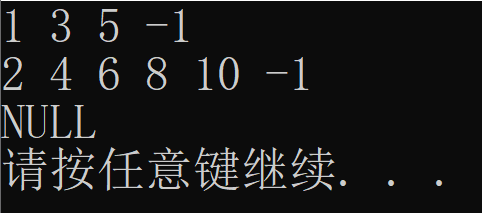
1 3 5 -1

2 4 6 8 10 -1

**预期结果：**

NULL

**实验结果：**



### 3.1.3 测试3

**测试用例：**

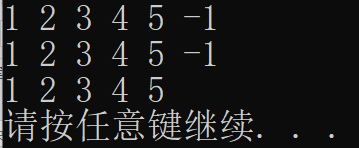
1 2 3 4 5 -1

1 2 3 4 5 -1

**预期结果：**

1 2 3 4 5

**实验结果：**



### 3.1.4 测试4

**测试用例：**

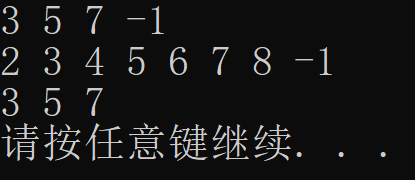
3 5 7 -1

2 3 4 5 6 7 8 -1

**预期结果：**

3 5 7

**实验结果：**



## 3.2 边界测试

### 3.2.1 其中一个链表为空

**测试用例：**

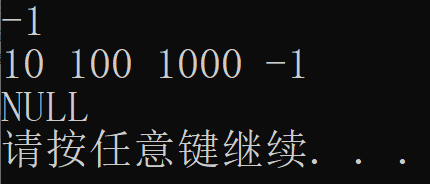
-1

10 100 1000 -1

**预期结果：**

NULL

**实验结果：**



### 3.2.2 两个链表为空

**测试用例：**

-1

-1

**预期结果：**

NULL

**实验结果：**

