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： 王星洲

学 号： 1652977

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](#_Toc531699426)

[1.1 背景分析 1](#_Toc531699427)

[1.2 功能分析 1](#_Toc531699428)

[2 设计 1](#_Toc531699429)

[2.1 数据结构设计 1](#_Toc531699430)

[2.2 链表的使用 2](#_Toc531699431)

[2.3 流程设计 2](#_Toc531699432)

[3 实现 2](#_Toc531699433)

[3.1链表的创建 2](#_Toc531699434)

[3.1.1 主要代码 2](#_Toc531699435)

[3.1.2 注意事项 3](#_Toc531699436)

[3.2 接收用户的输入 4](#_Toc531699437)

[3.2.1用户输入过程流程图 4](#_Toc531699438)

[3.2.2核心代码 5](#_Toc531699439)

[3.2.3 人机交互截屏示例 6](#_Toc531699440)

[3.2.4 人机交互优势及注意事项 7](#_Toc531699441)

[3.3 链表查找功能的实现 8](#_Toc531699442)

[3.3.1 查找功能流程图 8](#_Toc531699443)

[3.3.2 查找功能核心代码 9](#_Toc531699444)

[3.3.3 查找功能亮点及注意事项 9](#_Toc531699445)

[3.4 链表遍历功能的实现 10](#_Toc531699446)

[3.4.1 链表遍历功能流程图 10](#_Toc531699447)

[3.4.2 遍历功能核心代码 10](#_Toc531699448)

[3.4.3 遍历功能截屏示例 11](#_Toc531699449)

[3.4.4 遍历功能亮点及注意事项 11](#_Toc531699450)

[4 测试 11](#_Toc531699451)

[4.1 功能测试 11](#_Toc531699452)

[4.1.1 一般功能测试 11](#_Toc531699453)

[4.1.2 交集为空功能测试 12](#_Toc531699454)

[4.1.3 完全相交的功能测试 12](#_Toc531699455)

[4.1.4序列完全交集功能测试 13](#_Toc531699456)

[4.2 边界测试 13](#_Toc531699457)

[4.2.1 空集功能测试 13](#_Toc531699458)

[4.3 出错测试 13](#_Toc531699459)

[4.3.1 输入数据非法 13](#_Toc531699460)

[4.3.2 每行结尾不输入-1 14](#_Toc531699461)

[4.3.3 输入没有按照非降序输入 14](#_Toc531699462)

[5 亮点 14](#_Toc531699463)

[5.1查找算法亮点 14](#_Toc531699464)

[5.2人机交互亮点 14](#_Toc531699465)

# 1 分析

## 背景分析

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。

## 1.2 功能分析

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。

1. 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 |

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该程序主要需要依赖链表来解决问题。我采用一个链表来接收第一行数据，第二行数据在输入时就与链表中的数据进行比对，如果找到了该数据，就将该数据存储到结果链表中，最后判断结果链表是否为空，如果为空输出NULL，不空则遍历链表，输出。

## 2.2 链表的使用

对于本题，我采用#include<list>来使用STL库提供的标准链表，并使用自己编写的searchInList()函数，来对链表中的数据进行查找,并将结果添加到结果链表。

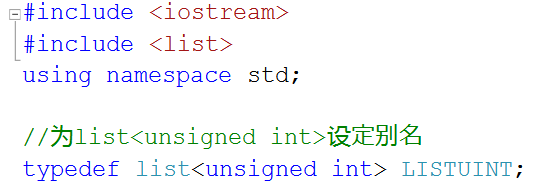
## 2.3 流程设计

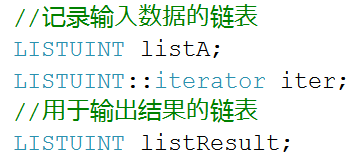
本程序首先创建接收数据用的链表A和输出结果用的链表Result。接下来读取用户的第一行输入，当用户输入“-1”时，输入完毕，并将数据保存在链表A中，接下来在用户输入第二行数据时，将每一个数据在A中使用searchInList()函数查找，如果查询到，则说明该数据是S1S2的交集，并将结果存入Result链表。输入完成后，检查Result链表是否为空，为空则输出”NULL”，非空则将链表遍历，输出结果。

# 3 实现

## 3.1链表的创建

### 3.1.1 主要代码





### 3.1.2 注意事项

1.注意包含头文件<list>

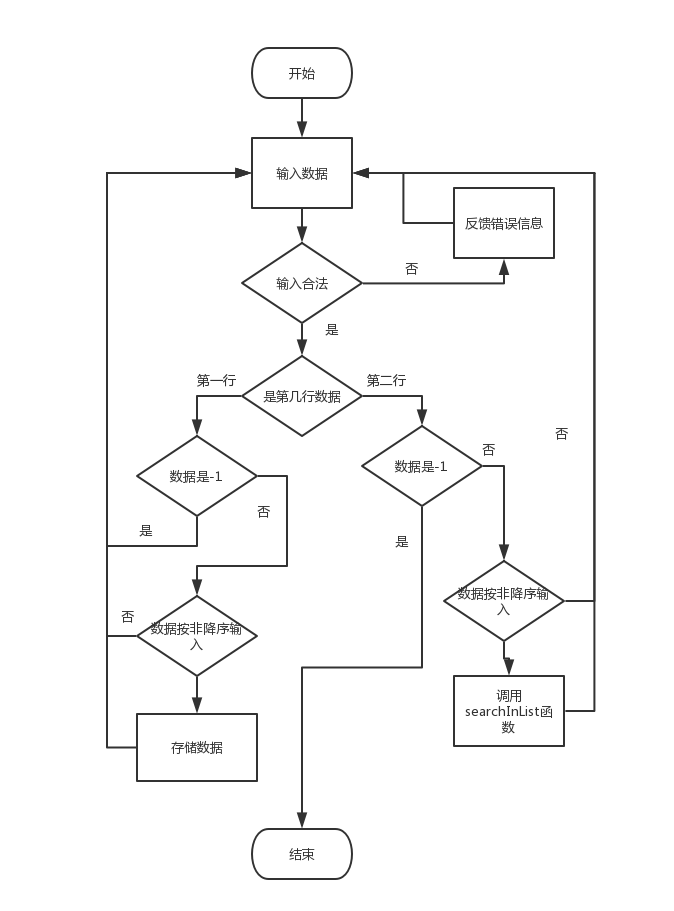
2.使用typedef为list<unsigned int>设置别名可以简化代码

3.因为题目中表明数据为正整数，所以使用unsigned节省空间

4.为listA设置一个迭代器iter为日后的查找做准备。

## 3.2 接收用户的输入

### 3.2.1用户输入过程流程图



### 3.2.2核心代码

//接收输入的值

int temp;

int temp1;

//接收输入，分为两行

for (int i = 0; i < 2; i++)

{

temp1 = -1;

//判断输入合法性

while (1)

{

cin >> temp;

if (!cin)

{

cout << "输入错误，请重新输入" << endl;

i -= 1;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

break;

}

//对于第一行输入

if (i == 0)

{

//输入-1时停止接收数据

if (temp == -1)

{

iter = listA.begin();

break;

}

else

{

if (temp1 < temp)

{

temp1 = temp;

//将数据存入listA

listA.push\_back(temp);

}

else

{

listA.clear();

cout << "请按照非降序输入，保证每个后输入的值大于前一个值。" << endl;

i -= 1;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

break;

}

}

}

//第二行数据

else

{

if (temp == -1)

{

break;

}

else

{

if (temp1 < temp)

{

temp1 = temp;

searchInList(temp, listA, listResult, iter);

}

else

{

listResult.clear();

cout << "请按照非降序输入，保证每个后输入的值大于前一个值。" << endl;

i -= 1;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

break;

}

}

}

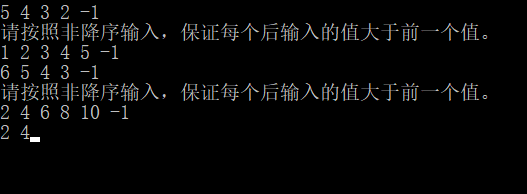
}

}

### 3.2.3 人机交互截屏示例







### 3.2.4 人机交互优势及注意事项

1.我在输入中添加了合法检测

cin >> temp;

if (!cin)

{

cout << "输入错误，请重新输入" << endl;

i -= 1;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

break;

}

因为temp是int类型，所以输入不合法的值会进入if语句，提示用户重新输入，同时将i-1，并且清空输入缓存区，更改cin标志，为下次输入做准备，这样做大大提升了用户的体验，防止程序出现问题。

2.同时输入中添加了非降序检测，如果用户没有保证输入的后一个值大于前一个，则给予用户提示信息，保证之后的查找算法不会出错。

else

{

listResult.clear();

cout << "请按照非降序输入，保证每个后输入的值大于前一个值。" << endl;

i -= 1;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

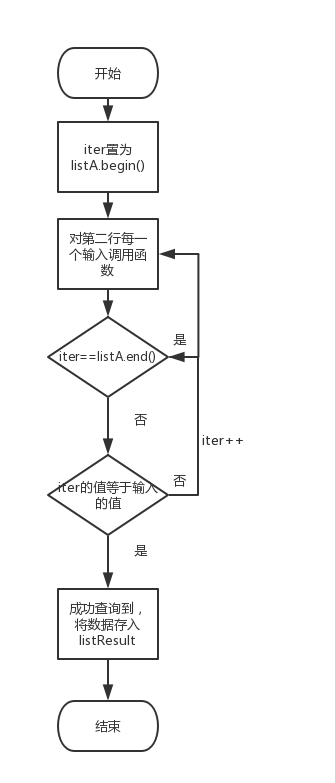
break; }

3.注意，检测合法代码中需要将输入流清空，否则将会在输出语句出现死循环

4.非降序检测中务必将对应链表清空。否则会让之前的错误数据污染链表。

## 3.3 链表查找功能的实现

### 3.3.1 查找功能流程图



### 3.3.2 查找功能核心代码

void searchInList(int temp, LISTUINT &listA, LISTUINT &listResult, LISTUINT::iterator& iter)

{

for (iter; iter != listA.end(); iter++)

{

if (temp == \*iter)

{

listResult.push\_back(temp);

return;

}

else

{

continue;

}

}

iter = listA.begin();

}

### 3.3.3 查找功能亮点及注意事项

1.本查找功能函数定义时务必使用引用传参，否则将会出现错误，拷贝复制的链表不可以使用该方法。错误提示为:Iterator is incompatible

2.本查找功能的亮点在于将iter也作为一个参数传入，在调用时从当前位置开始迭代，因为题目中给定的链表是非降序排列好的，所以只要上一次查询成功，就可以从上次的位置继续向下迭代：

举例说明:1 2 4 5

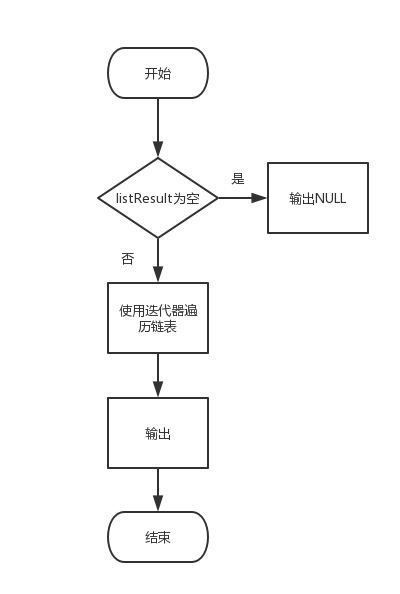
2 4 6 8 10

我们从链表二的2开始验证，当链表一迭代器从1迭代到2时，匹配，我们把2存入结果数组之后，下一次对链表二中的4进行匹配时，不需要从链表一的begin()，1开始迭代，而是从2的后一个数据，也就是4开始迭代，直接就可以匹配，这样做大大节省了时间成本。

当匹配失败时，迭代器会指向A的end()，这时不可以从当前位置开始，所以，匹配失败时，我们用iter = listA.begin();将迭代器重新定位到listA.begin()，避免出错。

## 3.4 链表遍历功能的实现

### 3.4.1 链表遍历功能流程图



### 3.4.2 遍历功能核心代码

void printList(LISTUINT listResult)

{

//如果listResult是空链表

if (listResult.empty())

{

//输出NULL，空格是为了格式

cout << "NULL ";

}

//listResult不空

else

{

//遍历listResult

for (LISTUINT::iterator iterResult = listResult.begin(); iterResult != listResult.end(); iterResult++)

{

//输出数据

cout << \*iterResult << " ";

}

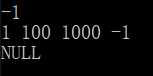
}

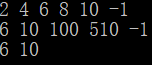
//清除多余的空格

cout << "\b";

}

### 3.4.3 遍历功能截屏示例





### 3.4.4 遍历功能亮点及注意事项

1.用list.empty()函数可以更容易判断链表是否为空

2.输出时，NULL和数据后都加一个空格，在最后使用\b清除可以实现题目中要求的格式化。

# 4 测试

## 4.1 功能测试

### 4.1.1 一般功能测试

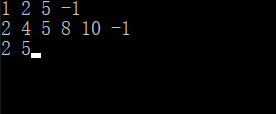
**测试用例**：

|  |
| --- |
| 1 2 5 -1  2 4 5 8 10 -1 |

**预期结果**：

输出2 5

**实验结果**



### 4.1.2 交集为空功能测试

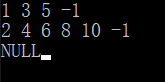
**测试用例：**

|  |
| --- |
| 1 3 5 -1  2 4 6 8 10 -1 |

**预期结果：**

NULL

**实验结果：**



### 4.1.3 完全相交的功能测试

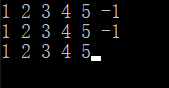
**测试用例：**

|  |
| --- |
| 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：**



### 4.1.4序列完全交集功能测试

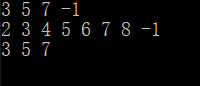
**测试用例：**

|  |
| --- |
| 3 5 7 -1  2 3 4 5 6 7 8 -1 |

**预期结果：**

3 5 7

**实验结果：**



## 4.2 边界测试

### 4.2.1 空集功能测试

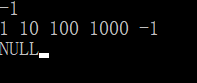
**测试用例：**

|  |
| --- |
| -1  1 10 100 1000 -1 |

**预期结果：**

**NULL**

**实验结果：**



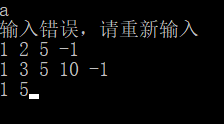
## 4.3 出错测试

### 4.3.1 输入数据非法

**测试用例：**输入字符而非数据

**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：**

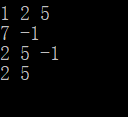


### 4.3.2 每行结尾不输入-1

**测试用例：**没有用-1结尾

**预期结果：**根据题目要求，我们视作本链表输入尚未完成，继续输入数据

**实验结果：**

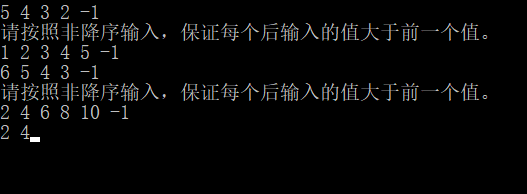


### 4.3.3 输入没有按照非降序输入

**测试用例：**数据顺序混乱

**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：**



# 5 亮点

### 5.1查找算法亮点

参考3.3.3

### 5.2人机交互亮点

参考3.2.3