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： 张诚睿

学 号： 2150998

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](#_Toc122644032)

[1.1 项目背景分析 1](#_Toc122644033)

[1.2 功能分析 1](#_Toc122644034)

[2 设计 1](#_Toc122644035)

[2.1 数据结构设计 1](#_Toc122644036)

[2.2 类结构设计 1](#_Toc122644037)

[2.3 成员与操作设计 1](#_Toc122644038)

[2.4 系统设计 2](#_Toc122644039)

[3 实现 3](#_Toc122644040)

[3.1 输入链表并建立的实现 3](#_Toc122644041)

[3.1.1 功能流程图 3](#_Toc122644042)

[3.1.2 核心代码 3](#_Toc122644043)

[3.1.3 输入界面效果展示 4](#_Toc122644044)

[3.2 求交集的实现 5](#_Toc122644045)

[3.2.1 功能流程图 5](#_Toc122644046)

[3.2.2 核心代码 5](#_Toc122644047)

[3.3 输出链表的实现 6](#_Toc122644048)

[3.3.1 功能流程图 6](#_Toc122644049)

[3.3.2 核心代码 7](#_Toc122644050)

[3.3.3 输出结果展示 7](#_Toc122644051)

[3.6 总体系统的实现 7](#_Toc122644052)

[3.6.1 总体系统流程 7](#_Toc122644053)

[3.6.2 总体系统核心代码 8](#_Toc122644054)

[3.6.3 总体系统截屏示例 8](#_Toc122644055)

[4 测试 8](#_Toc122644056)

[4.1 功能测试 8](#_Toc122644057)

[4.1.1 基本求交集功能测试 8](#_Toc122644058)

[4.2 边界测试 9](#_Toc122644059)

[4.2.1交集为空 9](#_Toc122644060)

[4.2.2 完全相交 9](#_Toc122644061)

[4.2.3 其中一个序列完全属于交集 10](#_Toc122644062)

[4.2.4 其中一个序列为空 10](#_Toc122644063)

[4.3 出错测试 11](#_Toc122644064)

[4.3.1 未输入-1结尾 11](#_Toc122644065)

# 1 分析

## 1.1 项目背景分析

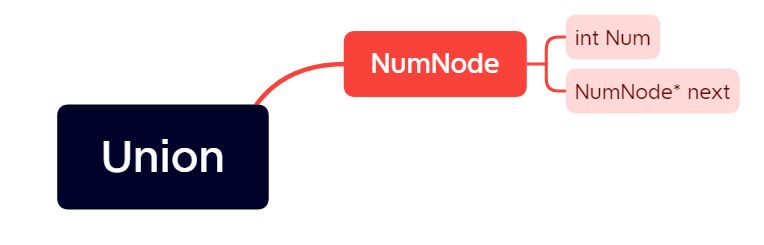
对于集合相关的数学分析过程，求解两个集合的交集是常用操作，而在实际操作中，数据量往往不确定且已完成排序方便操作，因此对两个有序不定长集合的排序是有需求的。

## 1.2 功能分析

本项的目的在于实现对两个有序不定长集合进行求交集操作并输出，要求输入的两个集合内容必须是已经排序完成的。

# 2 设计

## 2.1 数据结构设计



用Union类对NumNode结点进行链表管理，并实现初始化、求交集输出等操作

## 2.2 类结构设计

如上所述，本项目采用struct描述链表结点类（LNode），这样使得链表结点类（LinkList）可以访问链表结点。

## 2.3 成员与操作设计

**链表结点类**

struct NumNode {

int Num;

NumNode\* next;

};

**链表求交集类**

class Union {

private:

NumNode\* Ahead;

NumNode\* Bhead;

public:

Union();

void init(NumNode\*& head);

void output();

};

## 2.4 系统设计

首先建立Union对象，输入两个非降序数字序列并建立链表，之后求解交集至结果链表中，最后输出结果链表。

# 3 实现

## 3.1 输入链表并建立的实现

### 3.1.1 功能流程图

### 3.1.2 核心代码

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

{

int tmp = 0;

cin >> tmp;

if (tmp == -1)

{

break;

}

else

{

pointer = new(nothrow) NumNode;

if (pointer == NULL)

{

cout << "No Memory" << endl;

exit(EXIT\_FAILURE);

}

pointer->Num = tmp;

pointer->next = NULL;

}

if (i == 0)

{

head = pointer;

pointerRecord = pointer;

}

if (i != 0)

{

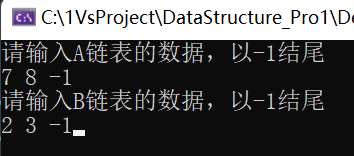
pointerRecord->next = pointer;

pointerRecord = pointer;

}

}

### 3.1.3 输入界面效果展示



## 3.2 求交集的实现

### 3.2.1 功能流程图

### 3.2.2 核心代码

while (p\_A != NULL && p\_B != NULL)

{

if (p\_A->Num == p\_B->Num)

{

if (tmp != p\_A->Num)

{

NumNode\* ans = new(nothrow) NumNode;

if (ans == NULL)

{

cout << "No Memory" << endl;

exit(EXIT\_FAILURE);

}

ans->Num = p\_A->Num;

ans->next = NULL;

p\_Out->next = ans;

p\_Out = p\_Out->next;

}

tmp = p\_A->Num;

p\_A = p\_A->next;

p\_B = p\_B->next;

}

else if (p\_A->Num > p\_B->Num)

{

p\_B = p\_B->next;

}

else if (p\_A->Num < p\_B->Num)

{

p\_A = p\_A->next;

}

}

## 3.3 输出链表的实现

### 3.3.1 功能流程图

### 3.3.2 核心代码

p\_Out = Output->next;

if (p\_Out == NULL)

{ // 说明未有交集

cout << "NULL" << endl;

return;

}

else

{

cout << p\_Out->Num;

p\_Out = p\_Out->next;

}

while (p\_Out != NULL)

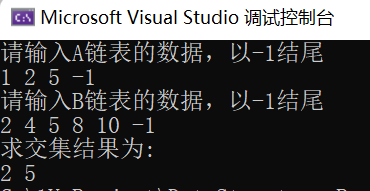
{

cout << " " << p\_Out->Num;

p\_Out = p\_Out->next;

}

### 3.3.3 输出结果展示



根据流程图可知，输出第一个数时直接打印，后续数字输出时先打印空格，再打印数字，达到数字之间有空格，且最后一个数后没有多余空格的效果。

## 3.6 总体系统的实现

### 3.6.1 总体系统流程

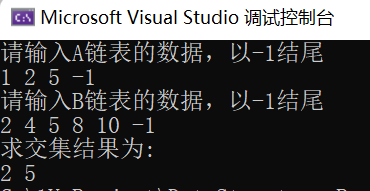
首先建立Union对象Intersection，随后调用Intersection.output()函数，完成所有操作后退出程序。

### 3.6.2 总体系统核心代码

Union Intersection;

Intersection.output();

### 3.6.3 总体系统截屏示例



# 4 测试

## 4.1 功能测试

### 4.1.1 基本求交集功能测试

**测试用例**：

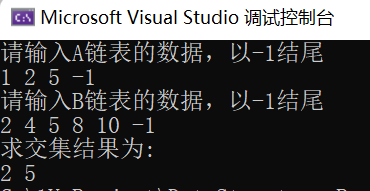
1 2 5 -1

2 4 5 8 10 -1

**预期结果**：

2 5

**实验结果**



## 4.2 边界测试

### 4.2.1交集为空

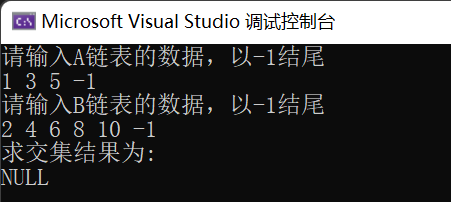
**测试用例：**

1 3 5 -1

2 4 6 8 10 -1

**预期结果：**NULL

**实验结果：**



### 4.2.2 完全相交

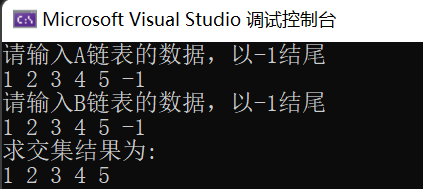
**测试用例：**

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：**



### 4.2.3 其中一个序列完全属于交集

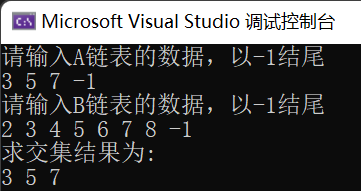
**测试用例：**

3 5 7 -1

2 3 4 5 6 7 8 -1

**预期结果：**3 5 7

**实验结果：**



### 4.2.4 其中一个序列为空

测试用例：

-1

10 100 1000 -1

预期结果：NULL

实验结果：



## 4.3 出错测试

### 4.3.1 未输入-1结尾

**测试用例：**1 2 3 4 5 6

**预期结果：**程序等待用户输入-1以结尾，不崩溃

**实验结果：**

