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： 肖杨

学 号： 1950430

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

2 设计

3 实现

1. 测试

1分析

1.1 项目要求分析

已知两个非降序序列S1和S2，设计构造函数构造出S1和S2的交集新链表。

1.2 功能分析

项目要求对链表进行操作，首先需要对链表进行建立，需要用到插入功能。构造出交集新链表之后要打印链表元素，需要用到输出功能。退出系统时要回收链表元素内存，需要用到清空功能。

综上所述，该考试管理系统需要有输入、输出、插入、清空的功能。

2 设计

2.1 数据结构设计

如上功能分析所述，项目需要使用链表的数据结构

2.2 类结构设计

对于本项目，未使用模板类，用直接的int型来设计Node和List类，用于简化操作，否则难以实现输入检测。Node储存数据和指针，List储存头尾结点，并提供插入、查找、删除等接口。之后在intersection类中储存两个List的指针与答案链表，并提供输出答案的接口。

本程序支持小于INT\_MAX的非降序排列链表求交集。

2.3 成员与操作设计

结点类

类定义：

class Node

{

friend class List;

friend class intersection;

protected:

int value;

Node\* next;

public:

Node(int val);

};

成员：

int value;

//整数值

Node\* next;

//指向下一个结点的指针

公有操作：

Node(int val);

//仅支持用值初始化，避免出现空结点

友元声明：

friend class List;

friend class intersection;

//保证List和intersection能对Node进行访问

链表类

类定义：

class List

{

friend class intersection;

public:

List();//构造函数

~List();//析构函数

void insert(const int &temp);//后端插入

int size();//链表长度

void printList();//输出链表

protected:

Node\* head;

Node\* present;

int length;

};

成员：

Node\* head;

//头结点

Node\* present;

//尾结点

int length;

//链表长度

公有操作：

List();

//默认构造函数，含头结点初始化

void insert(const int &temp);

//在链表最后端插入

int size();

//返回链表长度

void printList();

//输出链表，方便显示链表与输出答案

intersection类

类定义：

class intersection

{

public:

intersection();//构造函数

~intersection();//析构函数

void printAnswer();//输出答案

void getNumber(int& temp);//输入数据与输入检测

private:

List\* first;

List\* second;

List\* answer;

};

私有成员：

List\* first;

//第一个链表

List\* second;

//第二个链表

List\* answer;

//结果链表

公有操作：

intersection();

//构造函数，内部直接输入两个链表并求解，得到结果链表

void printAnswer();

//输出结果链表

void getNumber(int& temp);

//输入数据与输入检测

2.4 系统设计

系统会首先调用intersection类的构造函数，在构造函数内输入两个链表并求解，之后到用intersection类的printAnswer函数，并给出正确交集答案。

3 实现

3.1 输入链表的实现

3.1.1 输入链表流程

首先，考虑数据的读入，选择使用函数实现来进行数据筛选。

同时，在输入时保证非降序排列，随后存入链表。

之后，会将读入的链表显示以确认。

3.1.2 核心代码

insertion():

int temp = 0;

cout << "请输入一行非降序排列的数字，以-1结尾"<<endl;

getNumber(temp);

while (temp != -1)

{

if ((first->size() == 0 && temp != 0) || (first->size() != 0 && first->present->value <= temp))

{

first->insert(temp);

}

else

{

cout << "请非降序排列，已跳过非法输入" << endl;

}

getNumber(temp);

}//完整读取第一个链表

cout << "读取到的第一组链表为：";

if (first->size())

{

first->printList();

}

else

{

cout << "NULL" << endl;

}

void intersection::getNumber(int& temp)

{

string checkIn;

cin >> checkIn;

if (checkIn == "-1")

{

temp = -1;

return;

}

if (checkIn.size() > 10)

{

temp = 0;

return;

}

for (auto elem : checkIn)

{

if (!(elem >= '0'&&elem <= '9'))

{

temp = 0;

return;

}

}

if (atoi(checkIn.c\_str())<INT\_MAX)

{

temp = stoi(checkIn);

}

else

{

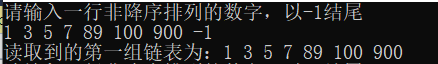
temp = 0;

}

}

//令temp为0，由于非降序排列，可以保证继续输入，并跳过非法输入

3.1.3 构造链表截屏示例



3.2 构造交集功能实现

3.2.1 构造交集流程

考虑到本身输入时已经是非降序排列，这里选择在构造第二个链表时直接进行比较，并得到交集。

利用一个指向第一个链表的指针，非空时持续将当前指针指向的值与新加入第二个链表的值进行比较。若相等则加入交集链表，并将指针指向下一个结点。否则，若大于当前值，则不变，若小于当前值，持续访问下一个结点直到找到值大于等于新加入值的结点，并进行一次比较判断是否相等。

之后，即得到完整的交集链表。

3.2.2 核心代码

Node\* presentptr=first->head;

presentptr = presentptr->next;//无论是否为空，都一定可以指向下一个

cout << "请输入一行非降序排列的数字，以-1结尾" << endl;

getNumber(temp);

while (temp != -1)

{

if (second->size() == 0 || (second->size() != 0 && second->present->value <= temp))

{

second->insert(temp);

if (presentptr != nullptr)//只有不为空才能进行判断

{

if (presentptr->value == temp)

{

answer->insert(temp);//相等直接得到答案

presentptr = presentptr->next;//当前值等于temp，指向下一个

}

else if (presentptr->value < temp)

{

while (presentptr != nullptr&&presentptr->value < temp)

{

presentptr = presentptr->next;//找到下一个大于等于temp的数字或读到尾端

}

if (presentptr != nullptr&&presentptr->value == temp)

{

answer->insert(temp);

presentptr = presentptr->next;//如果相等，执行上方相同代码

}

}//此时只有大于temp的不执行任何操作，直接读取下一个temp

}

}

else

{

cout << "请非降序排列，已跳过非法输入" << endl;

}

getNumber(temp);

}//此时所有链表均读取完毕

cout << "读取到的第二组链表为：";

if (second->size())

{

second->printList();

}

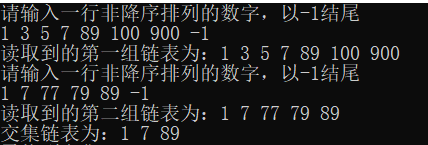
else

{

cout << "NULL" << endl;

}

3.2.3 构造交集功能截屏示例



3.3 总体功能实现

3.3.1 工作流程

首先调用intersection的构造函数，在构造函数中得到两个链表，并求出交集。

随后输出交集。

3.3.2 核心代码

int main()

{

intersection findIntersection;

findIntersection.printAnswer();

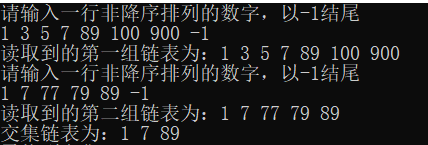
system("pause");

return 0;

}

大部分代码都在类中实现，故不重复贴出。

3.3.2 截屏示例



4 测试

4.1 基本功能测试

4.1.1 普通交集测试

**测试用例**：

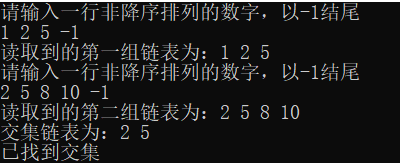
1 2 5 -1

2 5 8 10 -1

**预期结果**：

2 5

**实验结果：**



4.1.2 交集为空测试

**测试用例：**

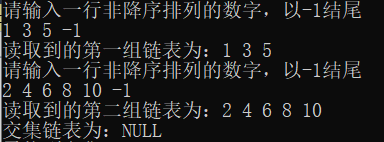
1 3 5 -1

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：**

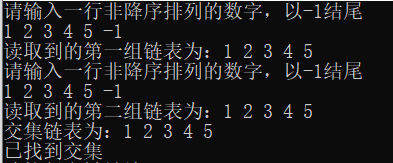
1 2 3 4 5 -1

1 2 3 4 5 -1

**预期结果：**

1 2 3 4 5

**实验结果：**



4.1.4 一个序列属于完全交集测试

**测试用例：**

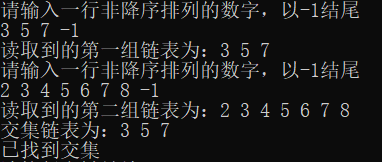
3 5 7 -1

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 一个序列为空

**测试用例：**

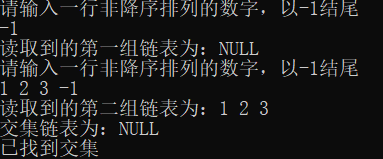
-1

1 2 3 -1

**预期结果：**

NULL

**实验结果：**



4.2.2 两个序列为空

**测试用例：**

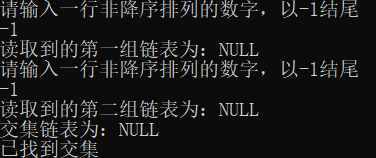
-1

-1

**预期结果：**

NULL

**实验结果：**



4.2.3 序列中数字极大(小于INT\_MAX)

**测试用例：**

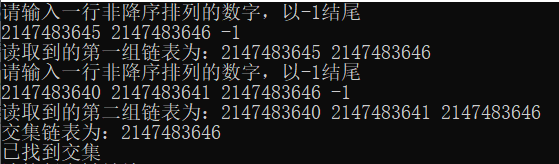
2147483645 2147483646 -1

2147483640 2147483641 2147483646 -1

**预期结果：**

2147483646

**实验结果：**



4.3 出错测试

4.3.1 输入内含有非数字

**测试用例：**

1 ASODJAWOIDJ111 2 3a 4 6 -1

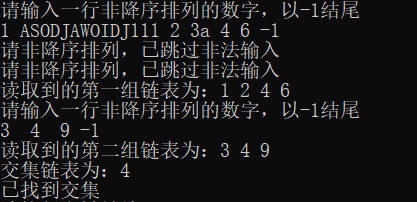
3 4 9 -1

**预期结果：**

考虑到输入不合法，给出提示并认为输入-1，将链表在非法输入处截断

4

**实验结果：**



4.3.2 输入序列中含有非-1负数

**测试用例：**

-22 -3 1 4 6 7 -1

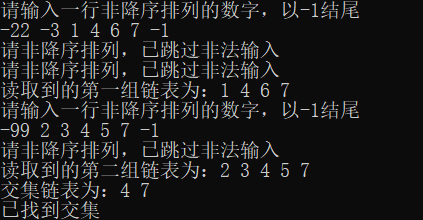
-99 2 3 4 5 7 -1

**预期结果：**

程序不崩溃，忽略异常输入，输出提示，正常求解，输出：

4 7

**实验结果：**



4.3.3 输入序列存在非降序部分

**测试用例：**

2 1 4 8 9 -1

4 1 9 8 10 -1

**预期结果：**

程序不崩溃，忽略非降序部分，输出提示，正常求解，输出：

4 9

**实验结果：**

