****

**Tongji University**

**数据结构课程设计实验报告**

**002-两个有序链表的交集**

专 业： 软 件 工 程

指导教师： 张 颖

学 号： 2 1 5 3 0 6 1

姓 名： 谢 嘉 麒

1. **项目简介**
   1. **项目综述**
      1. **项目内容**

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. **项目要求**

要求采用链表。

* + 1. **项目功能**

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. **项目文件整理**
3. intersection.h
4. intersection.cpp
5. main.cpp
   1. **项目各模块分析**
      1. **数据结构类型的选用**

考虑到本题要求求出两个有序链表的交集，故选用链表作为本次项目的数据结构；对于链表类的实现，为了提高泛用度，选择采用类模板的方式进行。

1. 建立带有附加头结点的单链表结点类：

template<class T>

class LinkNode {

public:

//将链表类声明为结点类的友元类，方便访问

friend class List<T>;

//重载输入输出运算符，并声明为结点类的友元函数

friend std::istream& operator>>(std::istream& istr, LinkNode<T>& x);

friend std::ostream& operator<<(std::ostream& ostr, const LinkNode<T>& x);

//仅初始化指针成员的结点类构造函数

LinkNode(LinkNode<T>\* ptr = NULL) { link = ptr; }

//初始化数据与指针成员的构造函数

LinkNode(const T& item, LinkNode<T>\* ptr = NULL) { data = item; link = ptr; }

private:

T data; //表内数据

LinkNode<T>\* link; //结点指针

};

1. 建立链表类：

template<class T>

class List {

public:

//链表类构造函数

List() { first = new LinkNode<T>; }

List(const T& x) { first = new LinkNode<T>(x); }

//复制构造函数

List(List<T>& L);

//置空链表

void makeEmpty();

//链表类析构函数

~List();

//计算链表的长度

int Length()const;

//返回附加头节点的地址

LinkNode<T>\* getHead()const { return first; }

//在链表中查找含有数据x的结点

LinkNode<T>\* Search(T x);

//搜索第i个元素的地址

LinkNode<T>\* locate(int i);

//取出第i个元素的值

bool getData(int i, T& x)const;

//用x修改第i个元素的值

void setData(int i, T& x);

//在第i个元素后插入x

bool Insert(int i, T& x);

//删去第i个元素，x返回该元素的值

bool Remove(int i,T& x);

//判断表是否空，空则返回true

bool IsEmpty()const

{

return first->link == NULL ? true : false;

}

//输出链表内所有结点

void outputAll();

//重载函数：赋值

List<T>& operator = (List<T>& L);

//后插法建立单链表

void inputRear(T endTag);

//前插法建立单链表

void inputFront(T endTag);

//求有序链表的交集

void intersection(List<T>& L);

private:

LinkNode<T>\* first; //单链表的表头指针

};

* + 1. **对于客户输入的错误处理**

本次项目要求客户手动输入内容，对于其可能出现的错误同样做了错误处理。依旧采用while循环+cin.函数的方式，此处不再赘述。

* + 1. **求两个有序链表的交集**

对于这一问题，笔者以链表类设定两个变量，先后提示客户进行内容的键入。客户确认后，将第二个链表变量作为参数代入第一个链表变量的求交集类成员函数中，并输出打印结果。

* + 1. **程序收尾工作的处理**

在完成有序链表的求交集操作后，提示客户操作完成，是否继续运算（y/n）：如果客户选择继续运算，则继续循环进行新一轮的运行；如果客户选择退出，则打印“欢迎下次使用”并退出循环，程序结束。对于每一次客户的键入，都需要进行输入错误处理。

1. **项目设计思路**

本次项目采用链表的方式进行实现，为了保证类定义的泛用性，采用了模板的方式。考虑到客户的输入与使用习惯，利用后插法建立单链表。此后，调用求交集函数进行运算，在main函数中进行测试，形成一套完整的求有序链表交集的项目。

* + 1. **客户端所需链表函数的思考**

本次项目再次使用001-考试报名系统中所采用的单链表的数据结构，但在此基础上将链表类用模板方式进行完善。通过对本项目要求与客户和程序之间的交互进行分析，本次程序需要调用后插法建立单链表的inputRear函数，以及求出并打印两个有序链表交集的intersection函数。其余提示输入和结束的交互模块和输入错误处理，可以在测试函数main中直接实现。

* + 1. **求两个有序链表交集的intersection函数**

本次项目将intersection设定为链表类的类成员函数，并且传入另一个链表类变量作为参数。求交集思路主要为以下几点：

1. 首先判断是否有链表为空；有，则交集为空；没有，则继续进行。
2. 其次当链表都不为空时，顺序比较两者数据是否相等；若相等，则都顺次比较下一个元素；若A链表元素小于B链表，则A链表当前指针指向下一个元素；若B链表元素小于A链表，则B链表当前指针指向下一个元素。
3. 设定count变量，若有元素相等则count++；若遍历完成后count依旧为初始值，则说明交集为空，打印NULL。
4. **主要功能的实现**
   1. **求链表交集函数**

本次项目功能重点落实在求有序链表的交集的实现。对于这一功能，可以参考求多项式的和这一数据结构经典案例。在这一基础上进行改进，优化，最终得到求两个有序链表交集的intersection函数如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称：List<T>::intersection(List<T>& L)

功 能：求两个有序链表的交集

输入参数：

返 回 值：

说 明：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

template<class T>

void List<T>::intersection(List<T>& L)

{

int count = 0;

LinkNode<T>\* p1, \*p2;

p1 = first->link;

p2 = L.first->link;

if (first->link == NULL || L.first->link== NULL) {

cout << "NULL" << endl;

return;

}

while (p1 && p2) {

if (p1->data == p2->data)

{

cout << p1->data << " ";

count++;

p1 = p1->link;

p2 = p2->link;

}

else if (p1->data < p2->data)

{

p1 = p1->link;

}

else

{

p2 = p2->link;

}

}

if (count == 0) {

cout << "NULL" << endl;

}

return;

}

* 1. **Main函数实现流程图**

提示客户输入内容

定义两个链表类变量list1,list2

Main函数

list1.intersection(list2)；

后插法建立单链表

客户是否继续

* 1. **Intersection函数实现流程图**

Intersection函数

两个链表中是否有一个为空

元素相等则指针“+1”

元素不相等则小元素指针“+1”

无相等元素则输出NULL

打印交集

1. **调试过程碰到的问题**
   1. **链表类变量之间的赋值运算符重载**

本次项目设定原计划将intersection函数设定为返回链表类变量，并且将此变量赋值给空闲变量list3。在以完成赋值运算符重载的情况下，笔者发现这种方式会导致编译器报错，故弃用。最终选择在intersection函数中进行交集的打印。

1. **心得体会**
   1. **对于数据结构知识的学习和掌握**

本次项目再次巩固了笔者对于单链表知识的掌握，也加深了笔者对链表各种操作的理解。同时也参考了多项式求和这一数据结构经典案例。经过本次项目的实现，笔者对于这些知识的掌握更加熟练，也提高了编程能力，优化了解题思路。

* 1. **完成本次作业的心得体会、经验教训**

1. 变量的使用、命名等需要易于理解，要直观简洁，反映其功能。
2. 对于全局变量的使用一定要谨慎，最好不用，即使不得已使用也要尽可能减少调用次数，限制适用范围。同时反复出现的常量最好用宏定义（或常变量）的形式在头文件中标明，既能方便调用，也便于后期的修补、维护。
3. 对于项目的理解一定要先整体把握，理清整体框架和大致思路，防止出现之后功能与之前功能断层的情况；在整体把握之后需要仔细整理细节，最好设置备忘录，保证程序细节处吻合。
4. 要充分思考，发散思维，在能力范围内寻找新思路、新方法，并从中选取优化部分。这一点不仅意味着要考虑代码可读性、繁琐性、是否满足细节需求，还应该 思考修补方案，在编码初期就为后期修补更正的工作留下空间。
5. 一定要写足够的注释！对不同函数功能、定义等进行分类，对不同参数、宏定义 等进行解读，能很大程度上辅助对代码的理解。
6. 要学会拆解大程序，构造小的测试空间，确保每一步编码的正确性，可以减少后 期出错后反复修改的成本
7. **程序测试**

****