同濟大學
TONGJI UNIVERSITY

项目说明文档

两有序链表的交集

指导教师：张颖

1751984 王舸飞

1.分析

1.1 背景分析

链表是一种较为基础的数据结构，能够正确有效的创造、遍历、比较链表是课程的基本要求，为此需要解决本次两有序链表的合成问题。

1.2 功能分析

本次设计的输入分为两行，以-1作为输入截止的标志，且规定输入为非降序，只需要逐次建立链表即可完成此输入操作。对于输出部分，要求用空格隔开且结尾不能有空格，此外，还应考虑到序列为空的特殊情况。

2.设计

2.1 数据结构设计

上述功能要求使用链表实现，故定义存储结构为链表。基于以前的经验，在链表之前添加一个附加头节点，此外，引入反应链表长度的变量length，用于方便的执行循环程序和防止访问为null的节点时编译器报错。

2.2 类结构设计

经典的链表一般包括两个抽象数据类型（ADT）——链表结点类（LNode）与链表类（LinkList），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。本程序中将链表节点类作为链表类的友元类，使得链表可以访问链表节点。

2.3 成员与操作设计

链表节点类（ListNode）
受保护的成员：

```
ListNode *next;       //链表的指针域

  int number;         //由于本次比较都为整数类型，故定义number为整形存储
```

公有操作：

ListNode(int p_number=0,ListNode *p_next=NULL){number=p_number;next=p_next;}
//构造函数，初始数字为0，下一个指针指向为空

int GetNumber(){return number;}　　//返回所存储的数字

ListNode* GetNext(){return next;}　　//返回下一个指针域

链表类（LinkedList）
受保护的成员：

ListNode *first;　//该链表的头指针

int length;　　//该链表的长度

公有操作：

LinkedList(int p_length=0){first=new ListNode;length=p_length;}
//构造函数，增加头节点并令长度为0

void InputData();　//初始化时输入数据所用函数

void Display()const;　//展示该链表

void Addnumber(int p_number);　//在末端插入数据p_number

ListNode* GetFirst(){return first;}.　//返回该链表的头节点

其他函数：

LinkedList Merge(LinkedList s1,LinkedList s2)　//将输入的两链表合并，返回新链表

char Compare(int a,int b) //比较a和b的大小，返回字符'>'、'<'或'='

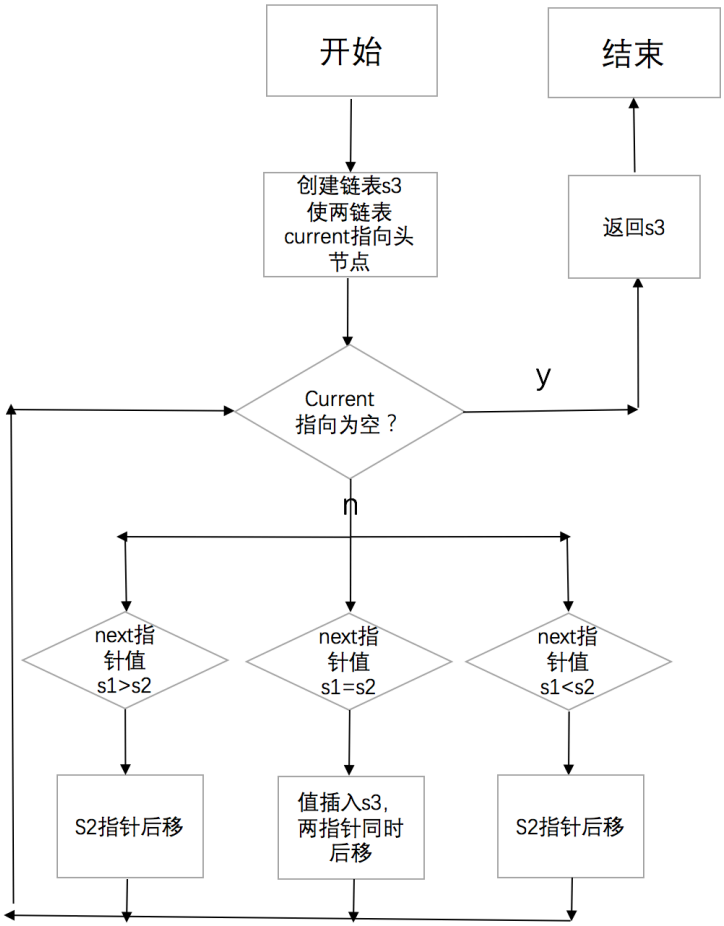
2.4 系统设计

程序运行之后，系统会创建三个链表s1、s2、s3，之后调用InputData()函数对s1、s2链表进行初始化赋值，之后调用Merge函数将两链表合并后的值放入s3中，最后将s3输出展示。

3.实现

3.1 合并功能的实现

3.1.1 合并功能流程图

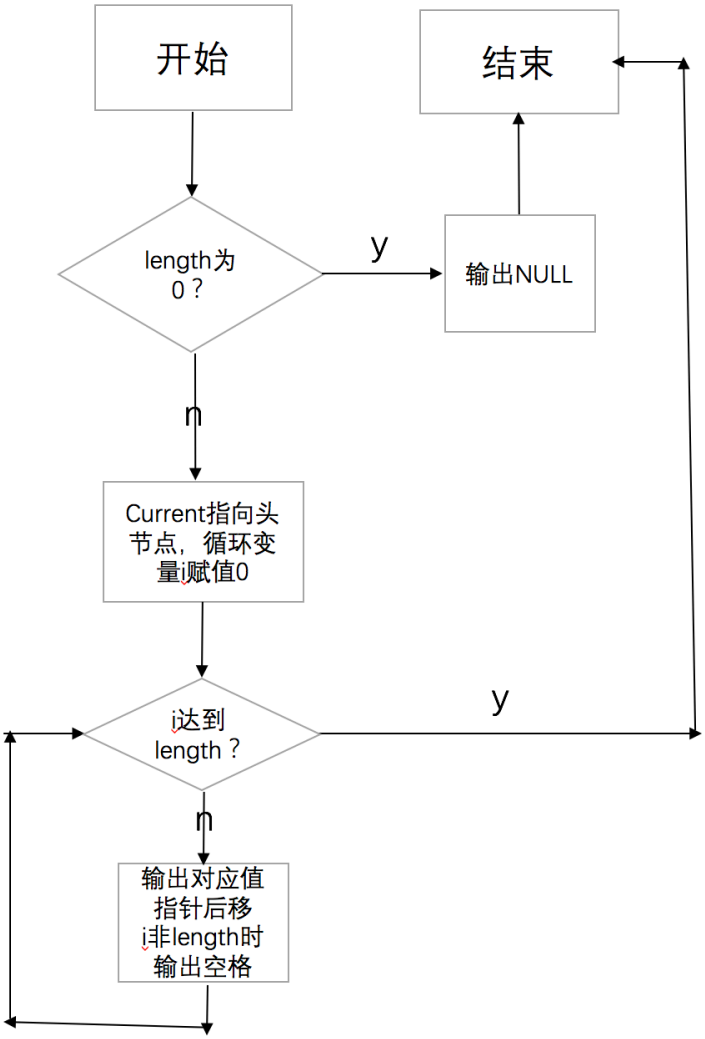


3.1.2 合并功能核心代码

```
LinkedList Merge(LinkedList s1,LinkedList s2){LinkedList s3;
    ListNode *s1_current=s1.GetFirst();ListNode *s2_current=s2.GetFirst();
    while(s1_current->GetNext()!=NULL&&s2_current->GetNext()!=NULL){
        switch (Compare(s1_current->GetNext()->GetNumber(),s2_current->GetNext()->GetNumber()))
        {case '>':s2_current=s2_current->GetNext();break;
            case '=':s3.Addnumber(s1_current->GetNext()->GetNumber());s1_current=s1_current->GetNext();s2_current=s2_current-
                >GetNext();break;
            case '<':s1_current=s1_current->GetNext();break;
        }
    }
    return s3;
}
```

3.1 展示功能的实现

3.1.1 展示功能流程图



3.1.2 展示功能核心代码

```
void LinkedList::Display()const{if(length==0)cout<<"NULL"<<endl;
    else{
    ListNode *current=first;
    for(int i=1;i<=length;i++){cout<<current->next->number;
        if(i!=length)cout<<" ";
        current=current->next;
    }
        cout<<endl;}
}
```

4.测试

4.1 功能测试

4.1.1 一般情况测试

测试用例：1 2 5 -1

2 4 5 8 10 -1

预期结果：2 5

实验结果：

```
1 2 5 -1
2 4 5 8 10 -1
2 5
Program ended with exit code: 0
```

4.1.2 交集为空的情况

测试用例：1 3 5 -1

2 4 6 8 10 -1

预期结果：NULL

实验结果：

```
1 3 5 -1
2 4 6 8 10 -1
NULL
Program ended with exit code: 0
```

4.1.3 完全相交的情况

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：

```
1 2 3 4 5 -1
1 2 3 4 5 -1
1 2 3 4 5
Program ended with exit code: 0
```

4.1.4 其中一个序列完全属于交集的情况

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 5 7 -1
2 3 4 5 6 7 8 -1
3 5 7
Program ended with exit code: 0
```

4.1.5 其中一个序列为空的情况

测试用例：-1

1 10 100 -1

预期结果：NULL

实验结果：

```
-1
1 10 100 -1
NULL
Program ended with exit code: 0
```