Tongji University

Data Structure Course Curriculum Design Document

Task2: Intersection of two ordered linked lists

Author:
嘉杰 李
Jiajie Li

Supervisor:
颖 张
Prof. Ying Zhang

A companion use documentation for a program submitted in fulfillment of the requirements for the Data Structure curriculum design

in the

School of Software Engineering, Tongji

December 27, 2019

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

Contents

Chapter 1

Analysis

1.1 Background Analysis

Seeking the intersection of two ordered linked list sequences is an important application of the data structure of a linked list. It has very important applications in information comparison and verification.

1.2 Functional Analysis

The input is divided into 2 lines, and each line gives a non-descending sequence consisting of several positive integers. Use -1 to indicate the end of the sequence (-1 does not belong to this sequence), and the numbers are separated by spaces. In the output line, the intersection sequence of the two input sequences is output. The numbers are separated by spaces. There must be no extra spaces at the end. If the new linked list is empty, the output is NULL.

Chapter 2

Design

2.1 Data structure design

As described in the functional analysis above, consider using a linked list data structure.

At the same time, in order to achieve simplicity, a head node is added before the first node, so that adding or deleting head nodes is the same as processing other nodes, and the program is concise.

2.2 Class structure design

I implemented the template linked list myself according to the SGI_STL linked list standard and placed it in ljj_linkedlist.h, including two abstract data types-linked list node class (___listNode) and linked list class (LinkList), and the coupling relationship between the two classes Multiple relationships such as nesting and inheritance can be used. In order to facilitate processing, the system uses a template struct to describe the linked list node class (___listNode), so that the template linked list node class (LinkList) can access the linked list node.

2.3 Member function design

Datanode for data

```
class datanode{
   public:
      bool avail=true;
      int data;
      int pdata() {
         cout << data << " ";
         return 0;
      }
   };
```

ListNode Class:

```
template <class T> struct __listNode {
   __listNode<T>* prev;
   __listNode<T>* next;
   T data;
};
```

Linked List Class:

```
template <class T> class list{
   private:
      typedef __listNode<T> list_node;
      typedef list_node* link_type;
      link_type get_node(){
         link_type p=(link_type)malloc(sizeof(list_node));
         p->next=NULL;
         p->prev=NULL;
         return p;
      }
      void put_node(link_type p){
         p->prev=NULL;
         p->next=NULL;
         p=NULL;
         free(p);
      }
      link_type create_node(const T& x){
         link_type p = get_node();
         p->data = x;
         return p;
      }
      void destroy_node(link_type p){
         put_node(p);
      }
      void __insert(link_type p, const T &x){
         len++;
         link_type temp = create_node(x);
         p->next->prev = temp;
         temp->next = p->next;
         temp->prev = p;
         p->next = temp;
      }
      void __del(link_type p){
         len--;
         p->prev->next=p->next;
         p->next->prev=p->prev;
         destroy_node(p);
      }
      link_type loc(int pos){
         pos+=1;
```

```
        if (pos>len) return NULL;
        link_type temp = hnode;
        while (pos--) temp=temp->next;
        return temp;
    }
    link_type loc_by_id(int id){
        auto temp = hnode->next;
        while (temp!=hnode) {
            if (temp->data.id==id) return temp;
            temp=temp->next;
        }
        return NULL;
    }

public:
    list();
    ~list();
    int Class_test(T temp);
    int push_back(const T& x) {__insert(hnode->prev, x); return 0;}
    int empty();
    int plist();
    bool isEmpty() {return hnode==hnode->next;}

    int del(int pos);
    int find_by_id(int id);
    int del_by_id(int id); // delete a node in the linked list
    int insert(int pos, const T &x);
    int modify(int pos, const T &x);
    int modify_by_id(int id, const T &x);

    int len;
    link_type hnode; //head node
};
```

2.4 System design

This program first defines three linked lists a, b, and c, and then calls the checklist function. In this function, first enter the data of the linked lists a and b, and then store the result of the intersection of the two linked lists into the linked list c.

Chapter 3

Implementation

3.1 Finding intersection

3.1.1 Finding function core code

```
   int main(){
   list<datanode> list1, list2, list3;
   datanode temp;
   while (temp.data!=-1) cin>>temp.data, list1.push_back(temp);
   list1.del(list1.len-1);
   temp.data=0;
   while (temp.data!=-1) cin>>temp.data, list2.push_back(temp);
   list2.del(list2.len-1);
   //list1.plist();
   //list2.plist();
   while (!list1.isEmpty() && !list2.isEmpty()) {
      int h1=list1.hnode->next->data.data;
      int h2=list2.hnode->next->data.data;
      if ( h1 == h2) {
         //cout << h1 << " ";
         list3.push_back(list1.hnode->next->data);
         list1.del(0);
         list2.del(0);
      }
      if (h1 < h2) list1.del(0);
      if (h1 > h2) list2.del(0);

   }
   list3.plist();
   return 0;
}
```

3.1.2 Finding function flowchart

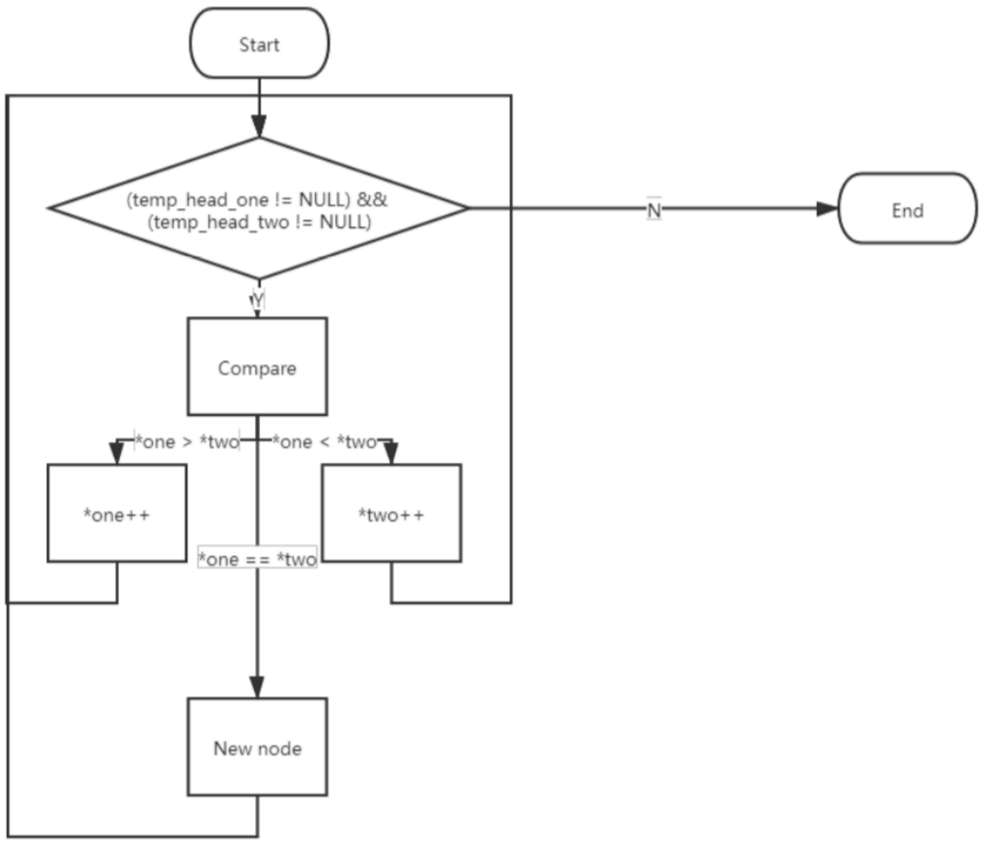


Figure 3.1: Finding function flowchart

Chapter 4

Test

4.1 Function Tests

4.1.1 Normal Situation

Test Case:

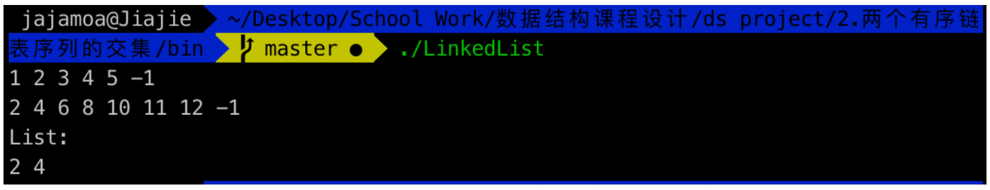
1 2 3 4 5 -1

2 4 6 8 10 11 12 -1

Expected Result:

2 4

Program Result:



```
 jajamoa@Jiajie  ~/Desktop/School Work/数据结构课程设计/ds project/2.两个有序链
表序列的交集/bin  master ●  ./LinkedList
1 2 3 4 5 -1
2 4 6 8 10 11 12 -1
List:
2 4
```

Figure 4.1: Screenshots

4.1.2 The intersection is NULL

Test Case:

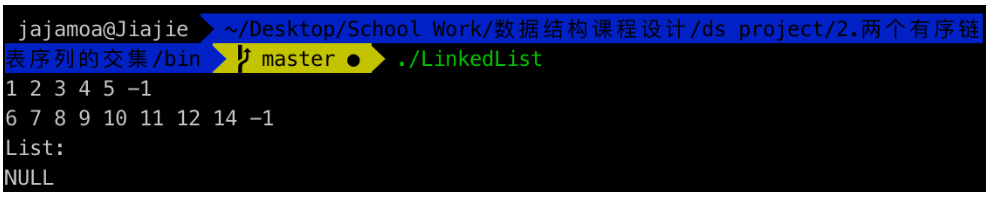
1 2 3 4 5 -1

6 7 8 9 10 11 12 14 -1

Expected Result:

NULL

Program Result:



```
 jajamoa@Jiajie  ~/Desktop/School Work/数据结构课程设计/ds project/2.两个有序链
表序列的交集/bin  master ●  ./LinkedList
1 2 3 4 5 -1
6 7 8 9 10 11 12 14 -1
List:
NULL
```

Figure 4.2: Screenshots

4.1.3 THe intersection is complete

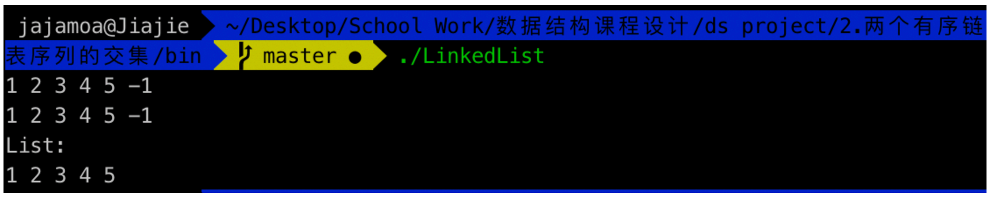
Test Case:

1 2 3 4 5 -1

1 2 3 4 5 -1

Expected Result:

NULL



```
jajamoa@Jiajie  ~/Desktop/School Work/数据结构课程设计/ds project/2.两个有序链
表序列的交集/bin  master ●  ./LinkedList
1 2 3 4 5 -1
1 2 3 4 5 -1
List:
1 2 3 4 5
```

Figure 4.3: Screenshots

4.1.4 One of the list is the same as the intersection

Test Case:

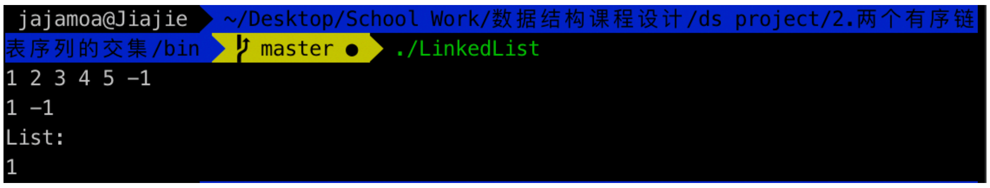
1 2 3 4 5 -1

1 -1

Expected Result:

1

Program Result:



```
jajamoa@Jiajie  ~/Desktop/School Work/数据结构课程设计/ds project/2.两个有序链
表序列的交集/bin  master ●  ./LinkedList
1 2 3 4 5 -1
1 -1
List:
1
```

Figure 4.4: Screenshots

4.1.5 One of the list is empty

Test Case:

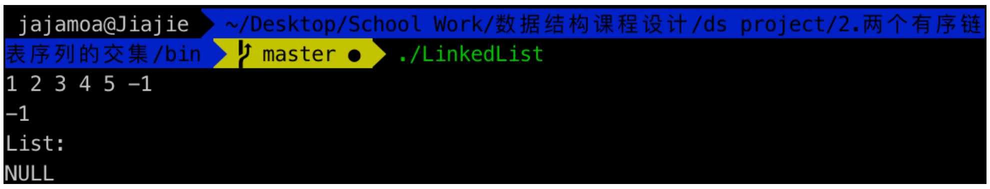
1 2 3 4 5 -1

-1

Expected Result:

NULL

Program Result:



```
 jajamoa@Jiajie  ~/Desktop/School Work/数据结构课程设计/ds project/2.两个有序链
表序列的交集/bin  master ●  ./LinkedList
1 2 3 4 5 -1
-1
List:
NULL
```

Figure 4.5: Screenshots

Appendix A

Frequently Asked Questions

A.1 Why this document looks so sparse?

This document is built using LaTeX and is arranged in book format. There may be blank pages before and after each chapter.

A.2 Why is there header files in this project?

I wrote basic stl header files, such as vector.h, etc. Except for the header files I wrote, the only header files used in all projects are iostream and string.h