专业：17电子信息工程

学号：17020022049

|  |  |
| --- | --- |
| https://ss1.bdstatic.com/70cFvXSh_Q1YnxGkpoWK1HF6hhy/it/u=2485631242,3638298595&fm=26&gp=0.jpg    **数据结构解题报告** | 曾慧敏 |

**目录**

**1、合并非降序链表---------------------------------------------------------2**

1.1实验目的-------------------------------------------------------------2

1.2实验内容-------------------------------------------------------------2

1.2.1基本步骤-------------------------------------------------------- 2

1.3实验输入和输出说明---------------------------------------------------2

1.3.1输入输出说明---------------------------------------------------- 2

1.3.2样例输出-------------------------------------------------------- 2

1.3.3输入样例的考虑因素---------- ------------------------------------2

1.4解题思路-------------------------------------------------------------3

1.5实验代码及注释-------------------------------------------------------4

1.6合并链表步骤流程图---------------------------------------------------5

1.7实验运行结果分析-----------------------------------------------------6

1.8实验收获总结---------------------------------------------------------7

**\_\_\_\_\_\_\_\_合并非降序链表\_\_\_\_\_\_\_\_**

**1、实验目的**

通过构造出两个非降序的链表序列M1和M2合并后的非降序链表，了解链表分类和特性，掌握链表构造及其应用方法。

**关键词**：链表 非降序 合并链表

**2、实验内容**

输入两个非降序的链表序列M1和M2作为已知链表, 构造出它们合并后的非降序链表。

基本步骤:

（1）定义链表结构

（2）构造合并链表函数

（3）构建链表输入程序（输入提示，考虑非法输入的情况）

（4）测试程序

**3、实验输入和输出说明**

**·Input**：两个非降序的链表序列M1和M2。（相邻节点元素值用“,”隔开，链表以“;”作为结束标志）

**·Output：**合并后的非降序链表（输出格式与输入链表的格式相同）

**·Example：**

**Sample Input:**

M1=1,2,4;

M2=1,3,4;

**Sample Output:**

M3=1,1,2,3,4,4;

**·测试中输入样例的考虑因素：**

1. **两个正常非降序列**

M1= 1,2,4;

M2= 1,3,4;

**（2）两个序列长度不相等**

M1= 1,3,4,10,19,20;

M2= 1,2,9,10;

**（3）序列中含有重复的数字**

M1= -12,0,1,1,3,4;

M2= 9,10,11,343,2090;

**（4）含有降序列**

M1= 0,3,5,2,1;

M2= 0,6,7,9,8,7,6;

**4、解题思路（时间复杂度O(n)）**

（1）在取空间最优的时候可能比较麻烦，故可以建立一个新的单链表M3，用来存放合并后的结果。

（2）遍历已知链表M1和M2的所有节点，每次都选取较小的那个元素值记录到M3中，同时小元素所在链表当前节点后移一位，继续比较，直到遍历完较短的链表。以保证M3中元素排列符合非降序的条件。

（3）由于M1,M2,M3都是按照非降序排列的，则当较短链表遍历完时，若M1或M2有元素剩余，则可以肯定这些元素均大于M3此时的所有元素，故直接将剩余部分连接到M3尾部即可

**5、实验代码及注释**

#include<iostream>

#include <string>

#include<stdlib.h>

using namespace std;

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* float val;

\* ListNode \*next;

\* ListNode(int x) : val(x), next(NULL) {}

\* };

\*\*/

//######### 定义单链表结构 ########

class ListNode

{

public:

int val;

ListNode \*next;

ListNode(int x) : val(x), next(NULL) {}

};

//########## 创建测试样例 ##########

ListNode \* create\_test\_case(){

ListNode \* l=new ListNode(0);

ListNode \* l\_head=l;

float val=0.0;

float pre=-12131;

char str=',';

while(str!=';'){

cin>>val>>str;

if(val<pre){

cout<<"invalid input!! the input has to be bigger than the previous one!!"<<endl;//考虑输入非法情况（即不按照非降序排列输入）

cout<<"but if we exchange the invalid element..."<<endl;

}

else{

pre=val;

ListNode \* M1= new ListNode(val);

l->next=M1;

l=l->next;

}

}

return l\_head->next;

};

//########## 合并序列 ##########

ListNode\* mergeTwoLists(ListNode\* M1, ListNode\* M2) {

ListNode\* head=new ListNode(0) ;//新建head作为合并后的序列

ListNode\* head1=head; //令head1为单链表head的头指针，作为对外接口

while(M1!=NULL && M2!=NULL){ //判断M1 val< M2 val是否成立

if(M1->val<=M2->val){

head->next=M1;

head=head->next;

M1=M1->next; //head序列连接较小元素

}

else{ //M1 val>M2 val

head->next=M2;

head=head->next; //head序列连接较小元素

M2=M2->next;

}

}

if(M1!=NULL)

head->next=M1; //判断所给两个链表是否还有剩余元素

else

head->next=M2;

return head1->next;

};

int main(){

cout<<"ATTENTION! elements of list have to be divided by ',' and end with ';'！"<<endl;//输入提示

cout<<"please input list M1:"<<endl;

ListNode\* M1=create\_test\_case();//输入第一个链表M1所包含元素

cout<<"please input list M2:"<<endl;

ListNode\* M2=create\_test\_case();//输入第二个链表M2所包含元素

ListNode\* M3=mergeTwoLists(M1,M2);

cout<<"the merged list M3 is:"<<endl;

while(M3->next!=NULL){

cout<<M3->val<<',';

M3=M3->next;

}

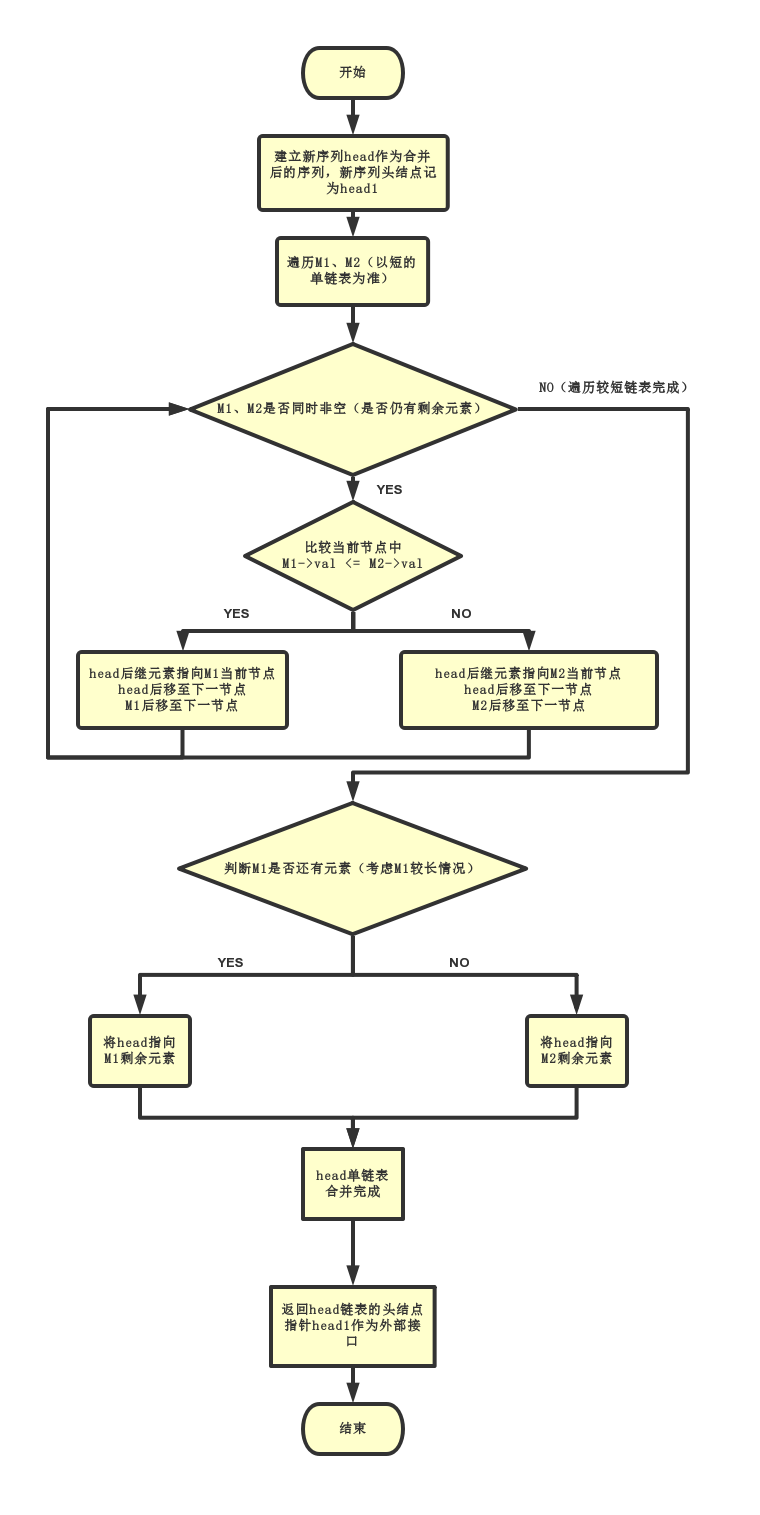
cout<<M3->val<<";"<<endl;

cout<<"\n"<<"\n"<<endl;

return 0;

}

**6、合并链表步骤流程图**

****

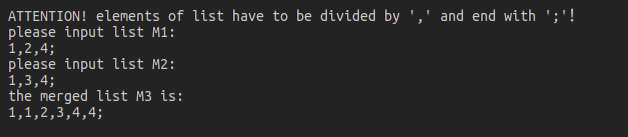
**7、实验运行结果分析**

**Input1:**

M1=1,2,4;

M2=1,3,4;

**Output1:**

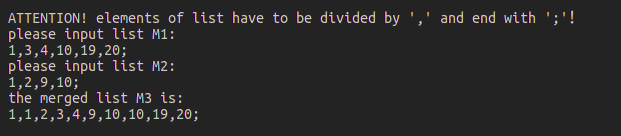


**Input2:**

M1=1,3,4,10,19,20;

M2=1,2,9,10;

**Output2:**

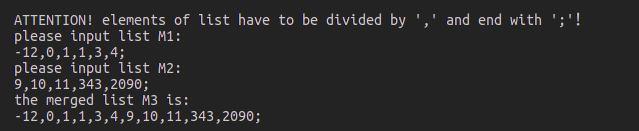


**Input3:**

M1=-12,0,1,1,3,4;

M2=9,10,11,343,2090;

**Output3:**

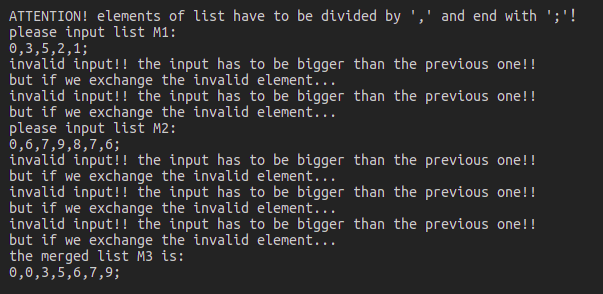


**Input4:**

M1=0,3,5,2,1;

M2=0,6,7,9,8,7,6;

**Output4:**



**8、实验收获总结**

**·在合并链表时如果为了不增加空间复杂度，则可以只在M1,M2的基础上进行增删节点的操作，但这样的操作注意要考虑到增删节点之后指针位置的变化。**

**·在进行增删节点操作时为简便起见，可以为链表添加一个头指针。**