线性表编程作业

姓名：寇一笑　　学号：18020024016　　姓名：安皓源　　学号：18020022001

2020 年 3 月 22 日

Contents

1 实验目的和内容

1.1 题目描述

已知两个非降序的双向链表序列M1和M2[1]，请构造出它们合并后的非降序双向链表，并分析算法的时间复杂度。

2 输入和输出说明

2.1 输入

第一行是以空格为分隔的非降序数字序列，结尾是-1表示结束　　第二行是以空格为分隔的非降序数字序列，结尾是-1表示结束

2.2 输出

输出一行以空格为分隔的非降序数字序列

2.2.1 样例

- 输入

  2 3 4 -1

  2 3 4 5 -1

- 输出

  2 2 3 3 4 4 5

- 输入

  -1

  -1

- 输出

  exit(1)

2.3 问题分析

根据题目，我们需要解决的问题有：

1. 如何建立双向链表
2. 如何输出双向链表
3. 如何将他们整合

[1]我们定义双向链表输入结尾是-1，这在题目中没有定义

3 解题思路

本道题主程序非常简单，主要是需要实现三个函数的功能。首先是创建一个双链表，这可以通过单链表加上一个prior指针来实现。首先建立一个头结点，判断输入的第一个输入是否为-1，如果是则返回exit(1)。然后每次输入一个数字判断是否为-1，如果不是则新建一个结点并给它赋值并且加上双向的链子。

这里最重要的函数是merge函数，为了节省空间，我们在其中一个链表La进行原地操作。先设置第三个链表Lc的头结点为La的，然后判断la和lb的大小，将小的插入到双向链表Lc中，如果相等，则先插入La的，再插入Lb的，实现非降序排列。

最后是输出函数，我们对加上了对于空链表输出NULL，当指针遍历不为*NULL*时，依次输出数据域的值。

4 实验代码及注释

4.1 双向链表版

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

typedef int ElemType;
typedef struct node
{                       //链表结点
    ElemType data;      //结点数据域
    struct node *next, *prior; //结点指针域
} Listnode, *Linklist; //节点类型，节点指针

Linklist create_list(); //此处用了单链表
Linklist merge_list(Linklist, Linklist);
void output_list(Linklist); //空链表将输出NULL

void main()
{
    Linklist L1, L2, L;
    printf("please input sequence1, end with -1:\t"); //本来想换行结尾，但是这样输入就是了char
    L1 = create_list();
    printf("please input sequence2, end with -1:\t");
    L2 = create_list();
    L = merge_list(L1, L2);
    printf("the result is:");
    output_list(L);
}

Linklist create_list() //创建链表并赋值
{
    int x;
    Linklist head, pa, pb; //头结点和两个指针,用来开辟空间，用来连接pbpapb
    scanf("%d", &x);
    if(x==1) exit(1); //如果第一个输入为，异常退出-1
    head = (Linklist)malloc(sizeof(Listnode)); //定义头结点
    pa = head;
    while (x != -1)
    {
```

```
        pb = (Linklist)malloc(sizeof(Listnode));
        pb->data = x;
        pa->next = pb;
        pb->prior = pa;
        pa = pb;
        scanf("%d", &x);
    }
    pb->next = NULL;
    return head; //返回头结点
}

Linklist merge_list(Linklist La, Linklist Lb)
{
    Linklist Lc, pa, pb, pc;
    Lc = La; //链表的头结点，直接在链表就地操作，降低空间复杂度ca
    pc = La;
    pa = La->next;
    pb = Lb->next;    //pa,pb,为指针pc
    while (pa && pb) //while (pa!=NULL && pb!=NULL) La,Lb 链表不为空
    {
        if (pa->data < pb->data) //为了实现非降序排列需要进行分类讨论
        {
            pc->next = pa;
            pa->prior = pc;
            pc = pa;
            pa = pa->next;
        }
        if (pa->data > pb->data)
        {
            pc->next = pb;
            pb->prior = pc;
            pc = pb;
            pb = pb->next;
        }
        else //相等的话，就重复上述两个操作
        {
            pc->next = pa;
            pa->prior = pc;
            pc = pa;
            pa = pa->next;

            pc->next = pb;
            pb->prior = pc;
            pc = pb;
            pb = pb->next;
        }
    }
    if (pa != NULL)
    {
        pc->next = pa;
    }
    if (pb != NULL)
    {
        pc->next = pb;
    }
    // 或者为空的话，直接将后续的链表加上去papb
    return Lc; //返回头结点
}
```

```
96
97  void output_list(Linklist s)
98  {
99      s = s->next;
100     if (s == NULL)
101     {
102         printf("NULL"); //空链表输出NULL
103         return;          //直接结束，不执行下面代码
104     }
105     while (s != NULL)
106     {
107         printf("%d ", s->data);
108         s = s->next;
109     }
110     printf("\n");
111     return;
112 }
113
```

Listing 1: 双向链表版本

4.2 双向循环链表版（未通过测试）

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <malloc.h>
4
5  typedef int Elemtype;
6  typedef struct node
7  {                       //链表结点
8      Elemtype data;      //结点数据域
9      struct node *next; //结点指针域
10 } Listnode, *Linklist; //节点类型，节点指针
11
12 typedef struct dnode
13 {
14     Elemtype data;              //结点数据域
15     struct dnode *prior, *next; //结点指针域
16 } DblNode, *DblList;             //节点类型，节点指针
17
18 DblList createDblList(); //此处用了双向循环链表
19 DblList mergeDblList(DblList, DblList);
20 void outputDblList(DblList); //空链表将输出NULL
21
22 void main()
23 {
24     DblList L1, L2, L;
25     printf("please input sequence1, end with -1:\t"); //本来想换行结尾，但是这样输入就是了char
26     L1 = createDblList();
27     printf("please input sequence2, end with -1:\t");
28     L2 = createDblList();
29     L = mergeDblList(L1, L2);
30     printf("the result is:");
31     outputDblList(L);
32 }
33
```

```
DblList createDblList() //建立双向循环链表
{
    DblList head, pa, pb;
    int x = 0;
    scanf("%d", &x);
    if (x == -1)
        exit(1);
    head = (DblList)malloc(sizeof(DblNode));
    head->prior = head->next = head; //表头结点的链指针指向自己
    pa = head;
    while (x != -1)
    {
        pb = (DblList)malloc(sizeof(DblNode));
        pb->data = x;
        pb->prior = pa;
        pb->next = pa->next;
        pa->next->prior = pb;
        pa->next = pb;
        pa = pb;
        scanf("%d", &x);
    }
    return head; //返回头指针
}

DblList mergeDblList(DblList La, DblList Lb)
{
    DblList Lc, pa, pb, pc;
    // Lc = (DblList)malloc(sizeof(DblNode));
    Lc = La; //链表的头结点，直接在链表头结点就地操作，降低空间复杂度ca
    pc = La; // 是用于插入的前一个结点pc
    pa = La->next;
    pb = Lb->next;                  //，是待插入的结点papb
    while (pa != La && pb != Lb) //while (pa!=NULL && pb!=NULL) //La,链表不为空Lb
    {
        if (pa->data < pb->data) //为了实现非降序排列需要进行分类讨论
        {
            pa->prior = pc;
            pa->next = pc->next;
            pc->next = pa;
            pc->next->prior = pa; //插入结点，并加入连接
            pc = pa;
            pa = pa->next; //后移，后移pcpa
            // DblList ptr = pa->next;
            // pa->prior = pc;
            // pa->next = pc->next;
            // pc->next->prior = pa;
            // pc->next = pa;
            // pc = pa;
            // pa = ptr;
        }
        if (pa->data > pb->data)
        {
            // DblList ptr = pb->next;
            // pb->prior = pc;
            // pb->next = pc->next;
            // pc->next->prior = pb;
            // pc->next = pb;
            // pc = pb;
```

```
92              // pb = ptr;
93              pb->prior = pc;
94              pb->next = pc->next;
95              pc->next = pb;
96              pc->next->prior = pb;
97              pc = pb;
98              pb = pb->next; //同上
99          }
100         else //相等的话，就重复上述两个操作
101         {
102             pa->prior = pc;
103             pa->next = pc->next;
104             pc->next = pa;
105             pc->next->prior = pa;
106             pc = pa;
107             pa = pa->next;
108
109             pb->prior = pc;
110             pb->next = pc->next;
111             pc->next = pb;
112             pc->next->prior = pb;
113             pc = pb;
114             pb = pb->next;
115         }
116     }
117     if (pa != La)
118     {
119         while (pa != La)
120         {
121             pa->prior = pc;
122             pa->next = pc->next;
123             pc->next = pa;
124             pc->next->prior = pa;
125             pc = pa;
126             pa = pa->next; //重复对操作a
127         }
128     }
129     if (pb != Lb)
130     {
131         while (pb != Lb)
132         {
133             pb->prior = pc;
134             pb->next = pc->next;
135             pc->next = pb;
136             pc->next->prior = pb;
137             pc = pb;
138             pb = pb->next; //重复对操作b
139         }
140     }
141     return Lc; //返回头结点
142 }
143
144 void outputDblList(DblList s)
145 {
146     DblList head = s;
147     s = s->next;
148     if (s == head)
149     {
```

```
150         printf("NULL"); //空链表输出NULL
151         return;         //直接结束，不执行下面代码
152     }
153     while (s != head)
154     {
155         printf("%d ", s->data);
156         s = s->next;
157     }
158     printf("\n");
159     return;
160 }
161
```

Listing 2: 双向循环链表版

5 运行结果截图

```
PS C:\c_algorithm> cd "c:\c_algorithm\" ; if ($?) { gcc merge_two_list_dbl.c -o merge_two_list_dbl } ; if ($?) { .\merge_two_list_dbl
 }
please input sequence1, end with -1:    2 3 4 -1
please input sequence2, end with -1:    2 3 4 5 -1
the result is:2 2 3 3 4 4 5
```

图 1: 运行结果

6 总结体会

第一次做这道题时我是用单链表做的，因为一开始没有看清题目的要求是双链表，成功的通过了测试，但不符合题目描述。之后我模仿着ppt使用双向循环链表，但是没有通过测试，debug结果为能够成功创建和输出双向链表，但是不能进行合并。于是在单链表上改成了双链表。这次因为电脑容量不足以安装*visualstudio*在配置*vscode*的*debug*配置json环境花了很多时间。

参考文献

[1] 邓俊辉. 数据结构（c++语言版）. 清华大学出版社.