数据结构 A
作业 3 参考答案

作业情况

教材：数据结构教程（C++ 语言描述）李春葆等
题目范围：线性表
邮箱：wjyyy1@126.com
授课教师：彭蓉 教授
助教：王骏峣

习题 1 集合并集运算

【问题描述】
给你两个集合 A 和 B，要求 $\{A\}+\{B\}$。注意同一个集合中不会有两个相同的元素。
【输入形式】
每组输入数据分为三行，第一行有两个数字 n，m（$0 < n, m \leq 10000$），分别表示集合 A 和集合 B 的元素个数。后两行分别表示集合 A 和集合 B，每个元素为不超出 `int` 范围的整数，两个元素之间有一个空格隔开。
【输出形式】
针对每组数据输出一行数据，表示合并后的集合，要求从小到大输出，两个元素之间有一个空格隔开。
【样例输入】

```
1 2
1
2 3
```

【样例输出】

```
1 2 3
```

【样例说明】
第一行的两个数字表示集合 A 和集合 B 的元素个数，后面两行分别表示集合 A 和集合 B 中的整数元素，两个元素之间用空格隔开。测试数据存放在 `in.txt` 文件中。
【评分标准】
共 10 个测试用例，每通过一个测试得 10 分。

答案：

```cpp
#include<iostream>
#include<fstream>
using namespace std;
struct LinkNode                 //单链表结点类型
```

```
{
    int data;            //存放int类型数据元素
    LinkNode* next;              //下一个结点的指针
    LinkNode():next(NULL) {}     //构造函数
    LinkNode(int d):data(d),next(NULL) {}
};
int main()
{
    LinkNode *ahead=new LinkNode();//链表a
    LinkNode *bhead=new LinkNode();//链表b
    int n,m,x;//输入操作符，如果为-1就停止
    ifstream fin("in.txt"); //使用文件流从in.txt读取数据
    fin>>n>>m;
    for(int i=1;i<=n;i++)//插入链表a
    {
        fin>>x;
        LinkNode *p=ahead;
        while(p!=NULL)
        {
            if(p->next!=NULL&&p->next->data<x)//说明还未找到比x大的位置
                p=p->next;
            else//找到比x大或链表末尾
            {
                LinkNode *t=new LinkNode(x);
                t->next=p->next;
                p->next=t;
                break;
            }
        }
    }
    for(int i=1;i<=m;i++)//插入链表b
    {
        fin>>x;
        LinkNode *p=bhead;
        while(p!=NULL)
        {
            if(p->next!=NULL&&p->next->data<x)//说明还未找到比x大的位置
                p=p->next;
            else//找到比x大或链表末尾
            {
                LinkNode *t=new LinkNode(x);
                t->next=p->next;
                p->next=t;
                break;
            }
        }
    }
    ahead=ahead->next;//把表头向后移动
    bhead=bhead->next;//将两组中最小的数据进行比较
    while(ahead!=NULL&&bhead!=NULL)
    {
        if(ahead->data==bhead->data)//相等时同时输出并后移
        {
```

```
58              cout<<ahead->data<<" ";
59              ahead=ahead->next;
60              bhead=bhead->next;
61          }
62          else if(ahead->data<bhead->data)//不相等时输出较小的一个
63          {
64              cout<<ahead->data<<" ";
65              ahead=ahead->next;
66          }
67          else
68          {
69              cout<<bhead->data<<" ";
70              bhead=bhead->next;
71          }
72      }
73      while(ahead!=NULL)//清空剩余的链表
74      {
75          cout<<ahead->data<<" ";
76          ahead=ahead->next;
77      }
78      while(bhead!=NULL)
79      {
80          cout<<bhead->data<<" ";
81          bhead=bhead->next;
82      }
83      return 0;
84  }
```

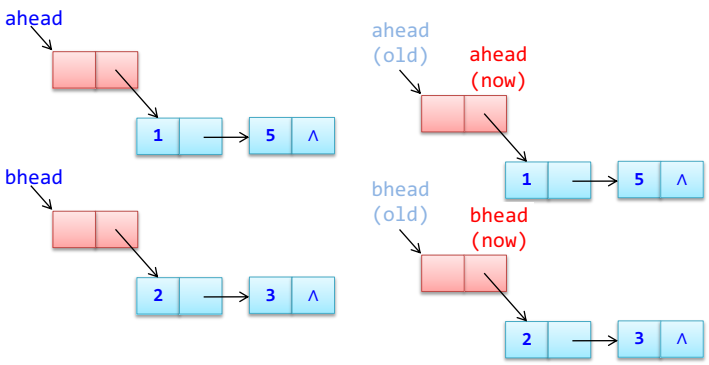
解析：参考课件第三讲中的 2.5.2 节例题，对输入的集合进行排序后，通过二路归并。

注意输入数据不一定是有序的，所以在插入的时候需要按顺序插入。或者对构建好的链表进行插入排序。

此外还可以使用 STL 中的 `list` 容器或 `vector` 容器，输入后使用 `sort()` 函数进行排序。排好序之后使用二路归并得到结果输出即可。

具体二路归并就是指将两个序列进行排序之后，每次找**两个序列的最小值中更小的那个**，并将其原先队列的指针后移一位。

而因为一开始头指针自身不包含 data 数据，需要将自己移到下一个位置才能开始进行处理，如下图所示。



习题 2 归并排序

【问题描述】

有一个含 n（$n \leq 200000$）个整数的无序序列，采用链表的二路归并排序实现递增排序。

【输入形式】

一行字符串，包含多个整数，每个数之间用空格分开。

【输出形式】

递增排序的结果，每个数之间用空格分开。

【样例输入】

```
9 4 7 6 2 5 8 1 3
```

【样例输出】

```
1 2 3 4 5 6 7 8 9
```

【样例说明】

测试数据的文件名为 in.txt，输出文件名为 out.txt。

【评分标准】

该题目有 10 个测试用例，每通过一个测试用例得 10 分

答案：

```cpp
#include<iostream>
#include<fstream>
using namespace std;
struct LinkNode              //单链表结点类型
{
    int data;            //存放int类型数据元素
    LinkNode* next;              //下一个结点的指针
    LinkNode():next(NULL) {}     //构造函数
    LinkNode(int d):data(d),next(NULL) {}
};
void merge_sort(LinkNode* head,int len)//传入链表头指针和长度，进行归并
{
    //如果len为1，则说明不需要再排序
    if(len==1)
        return;
    //将链表划分为两个长度相近的部分，将复杂度尽快减少
    int half=len/2;
    LinkNode* p=head;//将p定位到第half个元素
    for(int i=1;i<=half;i++)
        p=p->next;
    LinkNode* left_head=new LinkNode();
    LinkNode* right_head=new LinkNode();
    //让righthead指向第half+1个位置
    //并截断前面一半的链表
    left_head->next=head->next;
    right_head->next=p->next;
    p->next=NULL;
    //接下来对两个链表分别递归排序
```

```
    merge_sort(left_head,half);
    merge_sort(right_head,len-half);
    //两个子排序结束后，此时两个head指向的链表均为有序
    LinkNode* new_head=new LinkNode();
    LinkNode* new_tail=new_head;
    //接下来开始二路归并即可（参考上一题）
    left_head=left_head->next;//把表头向后移动
    right_head=right_head->next;//将两组中最小的数据进行比较
    while(left_head!=NULL&&right_head!=NULL)
    {
        if(left_head->data<=right_head->data)
        {
            new_tail->next=new LinkNode(left_head->data);
            new_tail=new_tail->next;//使用尾插法插入左边元素
            left_head=left_head->next;
        }
        else
        {
            new_tail->next=new LinkNode(right_head->data);
            new_tail=new_tail->next;//使用尾插法插入右边元素
            right_head=right_head->next;
        }
    }
    while(left_head!=NULL)
    {
        new_tail->next=new LinkNode(left_head->data);
        new_tail=new_tail->next;//使用尾插法插入左边元素
        left_head=left_head->next;
    }
    while(right_head!=NULL)
    {
        new_tail->next=new LinkNode(right_head->data);
        new_tail=new_tail->next;//使用尾插法插入右边元素
        right_head=right_head->next;
    }
    //最后将head指向new_head才能结束
    head->next=new_head->next;
}
int main()
{
    LinkNode *head=new LinkNode();//创建链表
    int n=0,x;//输入链表，统计长度
    ifstream fin("in.txt"); //使用文件流从in.txt读取数据
    ofstream fout("out.txt"); //使用文件流将数据输出到out.txt
    while(fin>>x)//使用头插法插入链表
    {
        LinkNode *p=new LinkNode(x);
        p->next=head->next;
        head->next=p;
        n++;
    }
    merge_sort(head,n);//递归进行归并排序
    head=head->next;
```

```
    while(head!=NULL)
    {
        fout<<head->data<<" ";
        head=head->next;
    }
    return 0;
}
```

解析：归并排序指的是将一段数列以中间为界限分为两部分，然后对两部分分别进行排序，最后将两部分合并。

归并排序分为三个步骤：

1. 将数列划分为两部分；2. 递归地分别对两个子序列进行归并排序；3. 合并两个子序列。

因为合并两个有序子序列的复杂度很低，所以通过归并排序可以在 $O(n \log n)$ 的时间内完成排序任务。

而此次作业我们输入的是链表，所以在对链表进行拆分的时候需要从中间断开，然后将两个表头传入函数，分别进行排序。

习题 3 归并排序

【问题描述】

有 M 个人，编号分别为 1 到 M，玩约瑟夫环游戏，最初时按编号顺序排成队列；每遍游戏开始时，有一个正整数报数密码 N，队列中人依次围坐成一圈，从队首的人开始报数，报到 N 的人出列，然后再从出列的下一人开始重新报数，报到 N 的人出列；重复这一过程，直至所有人出列，完成一遍游戏，所有出列的人形成新队列；游戏可能玩很多遍，每遍有新报数密码。求若干遍游戏完成后队列次序。本题要求使用单链表实现，程序要求采用模块化设计，格式规范，有合适注解。

【输入形式】

每个测试用例包含若干个正整数（至少 1 个），第一个正整数为玩游戏人数 M，后续每个正整数为每遍游戏报数密码；报数密码可能为 1。

【输出形式】

每个测试用例结果占一行，每个编号占 4 位。

【样例输入】

```
10 3 5 2
```

【样例输出】

```
   4   6   5   2   9   1   3   7   8  10
```

答案：

```
#include<iostream>
#include<fstream>
using namespace std;
struct LinkNode              //单链表结点类型
{
    int data;            //存放int类型数据元素
```

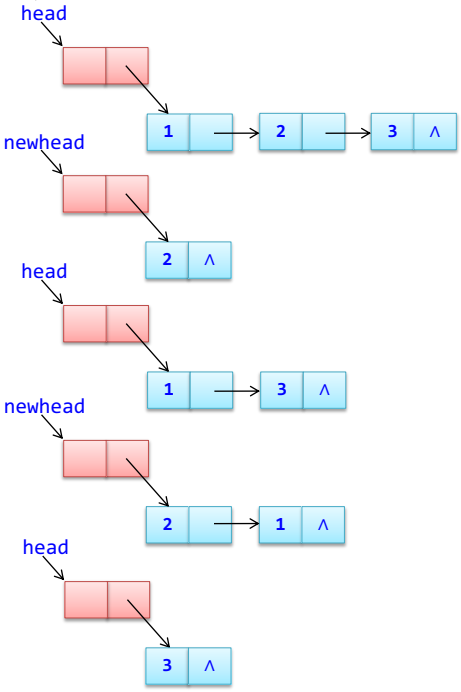
```
    LinkNode* next;              //下一个结点的指针
    LinkNode():next(NULL) {}     //构造函数
    LinkNode(int d):data(d),next(NULL) {}
};
int main()
{
    LinkNode *head=new LinkNode();//创建链表
    LinkNode *tail=head;//链表尾指针
    int m,n;//输入链表，统计长度
    cin>>m;
    for(int i=1;i<=m;i++)
    {
        tail->next=new LinkNode(i);
        tail=tail->next;//使用尾插法
    }
    while(cin>>n)//使用头插法插入链表
    {
        LinkNode *p=head;
        LinkNode *new_head=new LinkNode();
        LinkNode *new_tail=new_head;//创建新的环，仍使用尾插法
        while(head->next!=NULL)//说明环内仍有数字
        {
            //跳转n-1次
            for(int i=1;i<n;i++)
            {
                if(p->next!=NULL)//跳向p->next
                    p=p->next;
                else
                    p=head->next;
            }
            //跳n次之后p->next就是要被删掉的
            if(p->next!=NULL)//尾插法将p->next插入新链表
            {
                new_tail->next=new LinkNode(p->next->data);
                new_tail=new_tail->next;
                p->next=p->next->next;
            }
            else//然后删掉p->next
            {
                new_tail->next=new LinkNode(head->next->data);
                new_tail=new_tail->next;
                head->next=head->next->next;
            }
//              cout<<new_tail->data<<" ";
        }
        head=new_head;
//          cout<<endl;
    }
    head=head->next;
    while(head!=NULL)
    {
        cout<<setw(4)<<head->data;//保证宽度为4
        head=head->next;
```

```
60      }
61      return 0;
62  }
```

解析：本题需要模拟约瑟夫环，难点主要在于删除相应节点并拼接到新的链表上。

每次需要报数时就新建一个链表，报到第 n 个数字就删除当前指向的节点。因为要删除某个节点，所以应该保留“**要删除的点的**”上一个指针。此时把**要删除的点**用尾插法加入新的链表以便下一次报数。

图示如下：



总结

题目：线性表 2

日期：2024 年 4 月 5 日

批改人：王骏峣

邮箱：wjyyy1@126.com

习题 1：根据对题目意思的理解，对链表进行排序。然后合并两个有序链表需要用到归并的思想，这个较为容易（同时也出现在了第一次实验里，有细微不同）。

习题 2：延续习题 1 的归并思想，本题需要降低复杂度，关于归并排序的具体做法可以自行学习原理，或参考对数组的排序过程，最终写出链表的归并排序。部分同学对函数的理解不够具体，建议加强复习。

习题 3：注意本题是标准输入输出（不涉及文件），本地调试时可以阶段性输出来检验每次约瑟夫环的更新是否正确（参考答案中被注释掉的输出语句）。

在之后的作业中，如果具体实现方法与题目要求有出入但获得了分数（如使用链表的题目用了数组模拟），会酌情进行扣减，望大家注意。

各位同学如有问题欢迎及时在群里提出，或者通过邮件/QQ 联系我。