数据结构 A
期中考试参考答案与分析

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、栈、队列、串
邮箱：wjyyy1@126.com
授课教师：彭蓉 教授
助教：王骏峣

试题 1 链表重排

【问题描述】
给定一个单链表 $L_1 \to L_2 \to \cdots \to L_{n-1} \to L_n$，请编写程序将链表重新排列为 $L_n \to L_1 \to L_{n-1} \to L_2 \to \cdots$。例如：给定 L 为 $1 \to 2 \to 3 \to 4 \to 5 \to 6$，则输出应该为 $6 \to 1 \to 5 \to 2 \to 4 \to 3$。
【输入形式】
每个输入包含 1 个测试用例。每个测试用例第 1 行给出第 1 个结点的地址和结点总个数，即正整数 $N(\leq 10^5)$。结点的地址是 5 位非负整数，NULL 地址用 -1 表示。
接下来有 N 行，每行格式为：

```
Address Data Next
```

其中 Address 是结点地址；Data 是该结点保存的数据，为不超过 10^5 的正整数；Next 是下一结点的地址。题目保证给出的链表上至少有两个结点。
【输出形式】
对每个测试用例，顺序输出重排后的结果链表，其上每个结点占一行，格式与输入相同。
【样例输入】

```
00100 6
00000 4 99999
00100 1 12309
68237 6 -1
33218 3 00000
99999 5 68237
```

12309 2 33218
【样例输出】
68237 6 00100
00100 1 99999
99999 5 12309
12309 2 00000
00000 4 33218
33218 3 -1
【类库使用要求】
可以使用 STL 类库。
【输入类型】
标准输入
【输出类型】
标准输出
【评分标准】
10 个测试用例，按通过比例评分。
【解题思路】（仅供参考，可以有其他解法）
定义链表结点：包含三元组，*addr* 存储该结点本身的地址（可以定义为 int 或 string 类型，如果是 int 类型要注意位数不足时的输出时格式），*value* 存该结点序号，*next* 为指向下一结点的指针。
从输入数据构建链表，根据 *value* 值将其插入到链表相应位置；
按要求重排链表；
输出重排后的链表，输出内容为：“当前节点.addr”“当前结点.value”“下一结点.addr(尾节点输出 -1）”

答案：

```cpp
#include<iostream>
using namespace std;
int da[100000],ne[100000],node[100000];
bool renewed[100000];//表示这个地址的next被重置过
int main()
{
    int n,head;
    cin>>head>>n;
    for(int i=1;i<=n;i++)
    {
        int ad;//输入地址
        cin>>ad;
```

```
13          cin>>da[ad]>>ne[ad];
14          //将data和next放入相应地址
15      }
16      int now=head;
17      for(int i=1;i<=n;i++)
18      {
19          node[i]=now;
20          now=ne[now];//依次梳理L1L2L3分别是谁
21      }
22      //让Ln.next指向L1
23      //让L1.next指向Ln-1
24      //以此类推，Ln-i指向Li+1，Li指向Ln-i
25      int t=n,forward=0;
26      while(!renewed[t])//避免重复赋值
27      {
28          renewed[t]=1;
29          if(forward==1)
30          {
31              if(renewed[n-t])
32              {//如果发现下一个是找过的位置，就该停止了
33                  ne[node[t]]=-1;
34                  break;
35              }
36              ne[node[t]]=node[n-t];
37              forward=0;
38              t=n-t;
39          }
40          else
41          {
42              if(renewed[n-t+1])
43              {//如果发现下一个是找过的位置，就该停止了
44                  ne[node[t]]=-1;
45                  break;
46              }
47              ne[node[t]]=node[n-t+1];
48              forward=1;
49              t=n-t+1;
50          }
51      }
52      head=node[n];//从头（Ln）开始输出
53      while(head!=-1)
54      {
55          if(head<10000)
56              cout<<0;//不足5位补齐5位
57          if(head<1000)
58              cout<<0;
59          if(head<100)
```

```
            cout<<0;
        if(head<10)
            cout<<0;
        cout<<head<<" ";
        cout<<da[head]<<" ";
        if(ne[head]>=0&&ne[head]<10000)
            cout<<0;//注意-1不用补齐
        if(ne[head]>=0&&ne[head]<1000)
            cout<<0;
        if(ne[head]>=0&&ne[head]<100)
            cout<<0;
        if(ne[head]>=0&&ne[head]<10)
            cout<<0;
        cout<<ne[head]<<endl;
        head=ne[head];
    }
    return 0;
}
```

解析：题目要求将链表重新排列为 $L_n \to L_1 \to L_{n-1} \to L_2 \to \cdots$，所以要将每个位置的 *next* 进行修改，同时把表头放在最前面进行输出。

首先输入的链表不是按输入顺序排布的，需要先找到表头，然后依次梳理 $L_1, L_2, \cdots$。然后再将 L_n 放在最前，让其 *next* 指向 L_1（的地址），接着让 L_1 的 *next* 指向 L_{n-1}（的地址），以此类推。

所以本题需要把每个节点初始的编号找到，然后修改它的 *next* 指向即可。最后输出的时候要注意格式，一方面，每个节点的地址都是 5 位数，不足的要补零；另一方面要按照重排后的链表顺序 $L_n, L_1, \cdots$ 进行输出。

如果理解有问题可以考虑把上述代码的过程中的 *now* 和 *t* 逐个打印出来观察。

或直接参考题目给出的思路。

试题 2 超市模拟

（注：题目好像没有说读入输出方式，按标准输入输出）

【问题描述】

模拟超市排队的行为。最初，有 n 个队列（1，2，…，n），每个队列都有一些顾客。之后可能会发生两个事件：

ENTERS：顾客到达队列。如果队列在 1 到 n 之间，则顾客到达该队列的末尾。否则，该事件将被忽略；

LEAVES：顾客离开队列。如果队列在 1 到 n 之间，并且该队列不为空，则该队列的第一个顾客将离开该队列。否则，该事件将被忽略。

【类库使用要求】可以使用 STL 类库

【输入形式】
1. 输入从队列的数量 n（严格意义上为正的自然数）开始。
2. 按照 n 行，每个队列一行，每个行按照顾客到达队列的顺序，列出顾客名字。
3. 然后空一行。
4. 事件描述（ENTERS 或者 LEAVES）后面，跟着顾客名字和和队列序号（正整数）。
【输出形式】
1. 首先，按离开的顺序打印离开队列的顾客姓名。
2. 然后，按顺序打印 n 个队列的最终内容。
【样例输入】

```
5
Lisa Tom
John
Jerry Mary

Eric

LEAVES 1
LEAVES 2
ENTERS Harry 2
```

【样例输出】

```
DEPARTS
Lisa
John
FINAL CONTENTS
queue 1: Tom
queue 2: Harry
queue 3: Jerry Mary
queue 4:
queue 5: Eric
```

答案：

```cpp
#include<iostream>
#include<queue>
#include<sstream>
using namespace std;
```

```cpp
queue<string> q[100010];
int main()
{
    int n;
    cin>>n;
    string line;
    getline(cin,line);
    for(int i=1;i<=n;i++)
    {
        getline(cin,line);//读入一行
        istringstream in(line);//转为输入流
        string name;
        while(in>>name)
            q[i].push(name);
    }
    string op;
    cout<<"DEPARTS"<<endl;
    while(cin>>op)
    {
        if(op=="ENTERS")//入队
        {
            string name;
            int q_num;
            cin>>name>>q_num;
            if(q_num<=n&&q_num>=1)//需要判断是否合法
                q[q_num].push(name);
        }
        else
        {
            int q_num;
            cin>>q_num;//需要判断范围合法、队伍非空
            if(q_num<=n&&q_num>=1&&!q[q_num].empty())
            {
                cout<<q[q_num].front()<<endl;
                q[q_num].pop();
            }
        }
    }
    cout<<endl<<"FINAL CONTENTS"<<endl;
    for(int i=1;i<=n;i++)
    {
        cout<<"queue "<<i<<": ";
        while(!q[i].empty())//输出队伍内容
        {
            cout<<q[i].front()<<" ";
            q[i].pop();
        }
```

```
52          cout<<endl;
53      }
54      return 0;
55  }
```

解析：本题可以模拟若干个字符串（string）队列，但是注意输入时要将一整行转为一个字符输入流（istringstream），然后逐段读入。

此外，cin 会忽略空格和空行，如果需要读入空行或包含空格的一整行，需要使用 getline。

试题 3 栈处理数据序列

【问题描述】

从 txt 文件输入一个整数序列 $a_1, a_2, a_3, \ldots, a_n$，试编写算法实现，用栈结构存储输入的整数，当 $a_i \neq -1$ 时，将 a_i 进栈，当 $a_i = -1$ 时，使用 txt 文件输出栈顶整数并出栈，栈空间为 20，算法应对异常情况时输出错误信息（栈满输出 999、栈空下溢输出 -999 等）。

【类库使用要求】不允许使用 STL 中的容器类，比如 stack、vector、list 等，栈的操作需要自行编写代码。

【输入形式】

文件输入，文件名为 in.txt

【输出形式】

文件输出，文件名为 out.txt

【样例输入 1】

```
1 2 3 -1 -1 -1 -1
```

【样例输出 1】

```
3
2
1
-999
```

【样例输入 2】

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 1
```

【样例输出 2】

```
999
```

【样例输入 3】

```
1 2 3 4 -1 -1 -1 4 5 -1
```

【样例输出 3】

```
4
3
2
5
```

答案：

```cpp
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
    int s[25],tp=0,x;//tp指栈顶指针，tp=0表示为空
    ifstream in("in.txt");//输入流（用于替代cin）
    ofstream ou("out.txt");//输出流（用于替代cout）
    while(in>>x)
    {
        if(x==-1)
        {
            if(tp==0)//无法再出栈，栈空异常
            {
                ou<<"-999";
                return 0;
            }
            ou<<s[tp--]<<endl;//出栈同时弹出顶上元素
        }
        else
        {
            if(tp==20)//无法再入栈，栈满异常
            {
                ou<<"999";
                return 0;
            }
            s[++tp]=x;
        }
    }
    return 0;
}
```

解析：用数组模拟一个栈，用一个整型变量指向栈顶元素的下标，当这个下标为 0 时表示没有元素，即栈空；当这个下标为 20（栈空间大小）时，栈满。

栈空时如果要出栈，就是在让 0 接着减小，这是不合理的，输出 -999；栈满时如果要入栈，就是在让 20 接着增大，这也我们不想看到的，输出 999。

注：指向栈顶元素或栈顶元素的下一个位置都可以，但是要注意区分栈空和栈满时的情况。同时要注意数组要尽量开大一点，以免数组越界。而栈是否越界就由我们

代码里的逻辑来判断。

试题 4 将偶数移至奇数之前，保持相对次序不变

【问题描述】
从 txt 文件读入一个整数数组，数组长度在 0 到 200 之间，设计一个算法，将所有偶数移动到所有奇数的前面，要求它们的相对次序不变，使用 txt 文件输出移动后的结果。
【类库使用要求】不允许使用 STL 中的容器类，比如 queue、list、vector、stack 等，如用到队列、链表的操作需要自行编写代码。
【输入形式】
文件输入，文件名为 in.txt
【输出形式】
文件输出，文件名为 out.txt
【样例输入】
1 2 3 4 5 6 7 8
【样例输出】
2 4 6 8 1 3 5 7

答案：

```cpp
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
    int q0[205],cnt0=0;//q0按顺序存储偶数，cnt0存储偶数个数
    int q1[205],cnt1=0;//q1按顺序存储奇数，cnt1存储奇数个数
    ifstream in("in.txt");
    ofstream ou("out.txt");
    int x;
    while(in>>x)
    {
        if(x&1)
        {
            cnt1++;
            q1[cnt1]=x;
            //或合并为q1[++cnt1]=x;
        }
        else
        {
            cnt0++;
            q0[cnt0]=x;
```

```
23              //或合并为q0[++cnt0]=x;
24          }
25      }
26      for(int i=1;i<=cnt0;i++)
27          ou<<q0[i]<<" ";
28      for(int i=1;i<=cnt1;i++)
29          ou<<q1[i]<<" ";
30      return 0;
31  }
```

解析：实现两个队列，当输入数据是奇数则放入第二个队列，当输入数据是偶数则放入第一个队列。

最后先按顺序输出/弹出第一个队列，然后按顺序弹出第二个队列即可。

总结

题目：2023-2024 学年度第二学期期中考试

日期：2024 年 4 月 22 日

助教：王骏峣

邮箱：wjyyy1@126.com

期中测试大部分同学的问题出在输入输出上。第二次作业中我已经讲解过文件输入输出的基础方法，如果还是不太清楚的话可以直接记住以下几点：

- 标准输入输出的头文件是 `#include<iostream>`，文件输入输出的头文件是 `#include<fstream>`
- 标准输入输出为 `cin>>` 和 `cout<<`，箭头方向如果是输入就指向变量，如果是输出就从变量引出来，理解记忆。
- 当输入从标准输入变成文件输入时，就把 `cin>>` 换成我们自定义的 `in>>`。
  - **读入文件**：使用 `ifstream` 类，`ifstream in("in.txt")`，然后使用 `in>>x` 读入数据。
- 当输出从标准输出变成文件输出时，就把 `cout<<` 换成我们自定义的 `ou<<`。（这里不用 `out` 是担心 `out` 和标准库命名冲突）
  - **输出文件**：使用 `ofstream` 类，`ofstream ou("out.txt")`，然后使用 `ou<<x` 读入数据。

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。