

数据结构实验

一 . 实验要求

【问题描述】

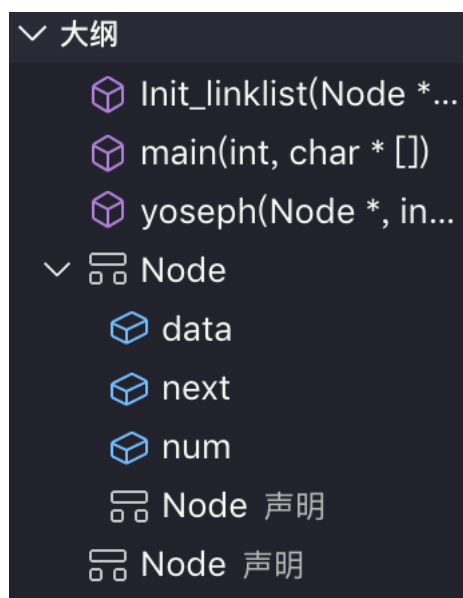
约瑟夫(Joseph)问题的一种描述是:编号为 $1, 2, \dots, n$ 的 n 个人按顺时针方向围坐一圈,每人持有一个密码(正整数)。一开始任选一个正整数作为报数上限值 m ,从第一个人开始按顺时针方向自 1 开始顺序报数,报到 m 时停止报数。报 m 的人出列,将他的密码作为新的 m 值,从他在顺时针方向上的下一个人开始重新从 1 报数,如此下去,直至所有人全部出列为止。试设计一个程序求出出列顺序。

1. 根据命令行输入的参数创建一个单向循环链表表示的 yoseph 环, 然后输出约瑟夫环出列的顺序 (基本要求)
2. 假设命令行参数是齐全的且正确的, 运行所编写的程序能正确输出结果 (基本要求)
3. 能将输出结果导入至文件中 (基本要求)
4. 程序有对命令行参数不全, 不正确的处理 (提示输入, 报错等) (选做要求)
5. 将 yoseph 环用顺序表表示 (选做要求)

二 . 设计思路

1. 链表法:

整体结构如下图:



其中 Node 结点中 data 元素用于存放每个人的密码，num 元素是每个人的序号，next 即为指向下一个结点的指针。

函数 Init_linklist 的声明如下：

```
struct Node* Init_linklist(struct Node* head,int n,int* input){  
    //利用命令行传递的参数，及刚刚申请的头指针创建链表(尾插法)，并且返回尾指针
```

由于链表并没有设置头结点且链表是循环的，所以我选择让构造函数返回一个尾指针，以便于之后的函数对链表中的结点进行删除。

n 是初始时的人数。

input 是 n 个人密码（int 型）组成的数组；main 函数中完成将字符串数组 agrv 转换为 input 的工作；若命令行中未输入参数，则用户直接将密码输入至 input 中。

函数 yoseph 的声明如下：

```
int *yoseph(struct Node *p, int n, int m)  
{  
    // 传入循环链表的尾指针 p，人数 n，初始报数间隔 m 每个人对应的密码已在链表中  
    //置好,返回一个 int 型数组 output
```

该函数是本次实验的核心函数，其返回值 output 即为指向输出结果的指针。

程序通过命令行传给 main 函数参数，main 函数构造 input 数组，之后调用 Init_linklist 函数构造链表，再通过 yoseph 函数得到 output

如果命令行中只有一个参数（程序名），则程序会继续执行且提示用户输入参数，具体实现代码如下：

```
if(argc>1)  
{  
    int k;  
    for(k = 0;k<n;k++)  
        input[k] = atoi(argv[i+3]);
```

```

        output = yoseph(Init_linklist(head,n,input), n, m);
    }
    else
    {
        int k = 0;

        printf("请依次输入%d 个人的密码 :",n);
        for(;k<n;k++)
        {
            printf("\n 第%d 个人的密码是 :",k+1);

            scanf("%d",input+k);
        }

        output = yoseph(Init_linklist(head,n,input), n, m);
    }
}

```

其中命令行中存在有效参数时($argc > 1$)进入 if 语句,使用字符串数组 argv 构造 input 数组; 当只有程序名一个参数时进入 else 语句, 提示用户输入参数以执行程序

最后在 main 函数中输出结果, 并写入文件“output00.txt”中,

```

for (i = 0; i < n; i++) //输出 output

    printf("%d ", output[i]);

//将 output 写入文件"output00.txt"中
FILE *fp = fopen("output00.txt","w");

if(fp == NULL){

    printf("文件打开错误\n");

    exit(-1);

}

for (i = 0; i < n; i++)

    fprintf(fp, "%d ", output[i]);

fclose(fp);

```

完毕。

2. 顺序表法(数组法):

整体结构如下图:



因为实现过程中并没有对表中元素进行删除操作, 所以并没有设计界沟通变量, 而直接使用了数组。

且本程序另外两个函数 `init_ring` 及 `yoseph` 和链表实现方法中的两个函数功能基本一致, 声明如下:

```
int *yoseph(int* p,int n,int m){  
    //传入的 p 是约瑟夫环顺序表, n 即为人数, m 为初始报数间隔  
  
status init_ring(int* p,int n,char* s){  
    //若构建成功返回 1, 失败返回-1
```

注意此程序, 不进行删除操作, 而是将某人的密码置为 0 以表明他已出列。

三 . 关键代码讲解

1. 链表实现:

下面为 `yoseph` 函数的代码:

```
int *yoseph(struct Node *p, int n, int m)  
{// 传入循环链表的尾指针 p, 人数 n, 初始报数间隔 m 每个人对应的密码已在链表中置好,返回一个 int 型  
数组 output  
    if (!p)
```

```

    return (NULL);

struct Node *s;

s = p->next;

int counter; //counter 记录被删去的人数

int *output = (int *)malloc(n * sizeof(int));

int i;

for (counter = 0; counter < n; counter++)
{
    m = m%(n-counter) == 0 ? (n-counter):m%(n-counter); //化简 m //n-counter 为表中现有人
数
    for (i = 1; i < m; i++)
    {
        p = p->next;
        s = s->next;
    }
    m = s->data;
    output[counter] = s->num;
    p->next = s->next;
    free(s);
    if (p)
        s = p->next;
}

return (output);
}

```

- (1) 采用循环删结点的方法。
- (2) 删除结点前将该结点的 num 值输入至 output 中，将其 data 值赋给 m 作为新的报数间隔
- (3) 为了方便对结点空间进行释放，设置了两个指针 p，s：其中 s 指向当前进行报数的对象，p 是其前一个指针以方便报数至 m 时对 s 进行删除
- (4) counter 是已经当前出列的人数，当 counter 等于 n 时循环

结束；且 n-counter 可代表队列中现有的人数。
(5)每次循环前对 m 进行了化简，以减小时间复杂度。

2. 顺序表实现：

yoseph 函数代码：

```
int *yoseph(int* p,int n,int m){
    //传入的 p 是约瑟夫环顺序表，n 即为人数，m 为初始报数间隔

    int i ; //i 用于判断是否达到了要求的间隔

    int j = 0; //j 作为 p 的偏移量

    int counter;

    int *output = (int *)malloc(n*sizeof(int));

    if(!output){

        printf("申请空间失败！\n");

        return(NULL);

    }

    for(counter = 0;counter<n;counter++){

        m= m%(n-counter) == 0 ? (n-counter):m%(n-counter); //化简 m

        for(i = 0;i<m;){

            j%=n;          //此时 j 的范围是 0 到(n-1)，p[j]有效

            if(p[j]) {i++;j++;}

            else    j++;

        }

        //此时 j 的范围是 1 到 n，p[j-1]有效

        output[counter] = j;

        m = p[j-1];

        p[j-1] = 0;

    }

    return (output);
}
```

其中各变量的功能在注释中已经简单给出。每当找到该出

列的人后就将其密码置为 0，相当于对其进行了标记，表明他已经出列。这样可以避免顺序表删除元素操作步骤多的缺点。

需要注意的是在队中序号是 j 的人在数组中的角标是 $j-1$ 。其它如 counter 的功能， m 的化简方法等,都与链表一致。

四 . 调试分析

1. 链表实现:

因为每次重新计数前，都要对 m 进行化简，所以每次最坏的情况下要将表中所有剩余结点都遍历一遍，平均每次有 $(n+1)/2$ 个结点，进行 n 次循环遍历，
时间复杂度为 $n*(n-1)/2 = O(n^2)$
用线性表存储，空间复杂度为 $O(n)$

实验中出现的問題：

最开始化简 m 时，直接将 m 模 $(n-\text{counter})$ 赋值给 m ，这导致了在 m 为剩余人数整数倍时被重新赋值为 0，是一种非法的报数间隔。之后对这一步重新设计，当出现上述情况时将 m 的新值赋为 $n-\text{counter}$ ，解决了该 bug.

2. 顺序表实现:

每次最坏的情况要将整个数组中元素全部遍历一遍，且不进行删除，每次遍历最坏情况都是 n 次，一共要进行 n 次遍历，
时间复杂度为 $n*n=O(n^2)$
空间复杂度显然为 $O(n)$

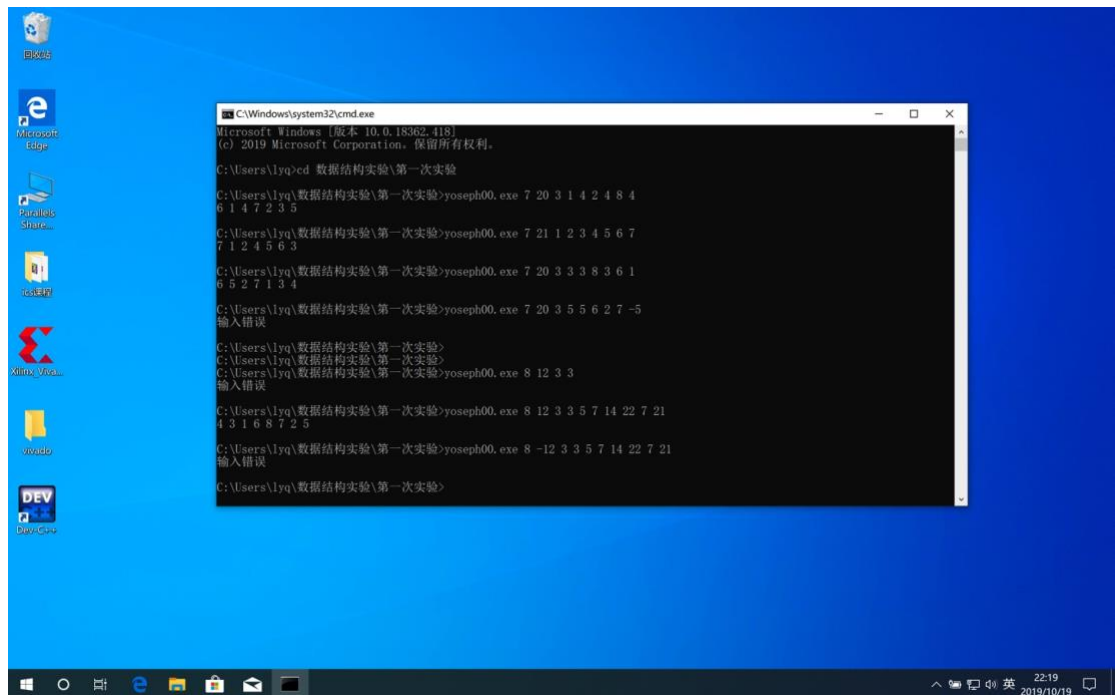
仿照链表实现编写了顺序表实现的程序，途中没有遇到太多问题。

五．代码测试

在 Windows10 操作系统下进行了测试，结果如下：

1. 顺序表：

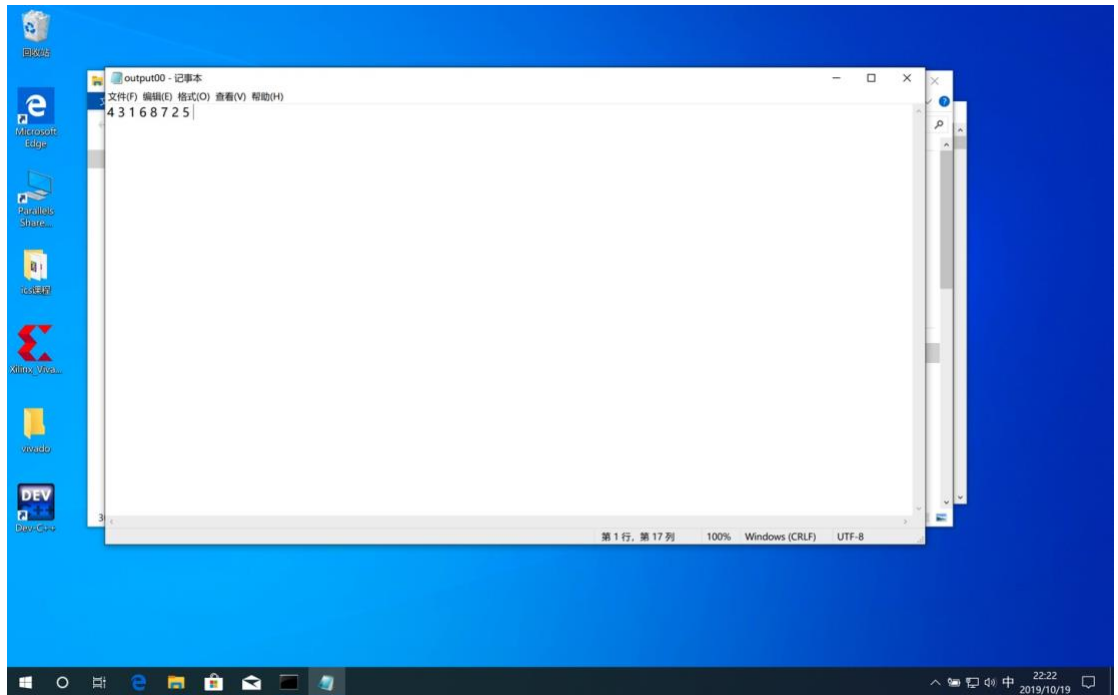
用多组输入（包含错误输入）进行测试：



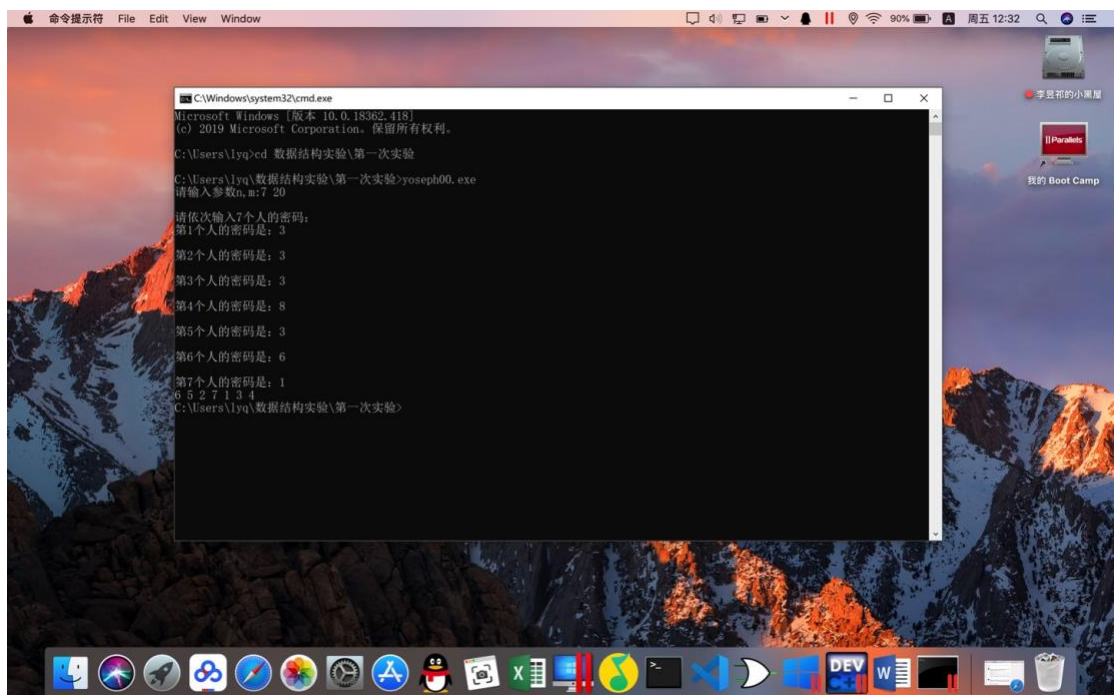
```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.18362.418]
(c) 2019 Microsoft Corporation. 保留所有权利。

C:\Users\lyq>cd 数据结构实验\第一次实验
C:\Users\lyq\数据结构实验\第一次实验>yoseph00.exe 7 20 3 1 4 2 4 8 4
6 1 4 7 2 3 5
C:\Users\lyq\数据结构实验\第一次实验>yoseph00.exe 7 21 1 2 3 4 5 6 7
7 1 2 4 5 6 3
C:\Users\lyq\数据结构实验\第一次实验>yoseph00.exe 7 20 3 3 3 8 3 6 1
6 5 2 7 1 3 4
C:\Users\lyq\数据结构实验\第一次实验>yoseph00.exe 7 20 3 5 5 6 2 7 -5
输入错误
C:\Users\lyq\数据结构实验\第一次实验>
C:\Users\lyq\数据结构实验\第一次实验>
C:\Users\lyq\数据结构实验\第一次实验>yoseph00.exe 8 12 3 3
输入错误
C:\Users\lyq\数据结构实验\第一次实验>yoseph00.exe 8 12 3 3 5 7 14 22 7 21
4 3 1 6 8 7 2 5
C:\Users\lyq\数据结构实验\第一次实验>yoseph00.exe 8 -12 3 3 5 7 14 22 7 21
输入错误
C:\Users\lyq\数据结构实验\第一次实验>
```

最后一次成功执行程序（合法输入）的 output 结果写入了文件 output00.txt 中：



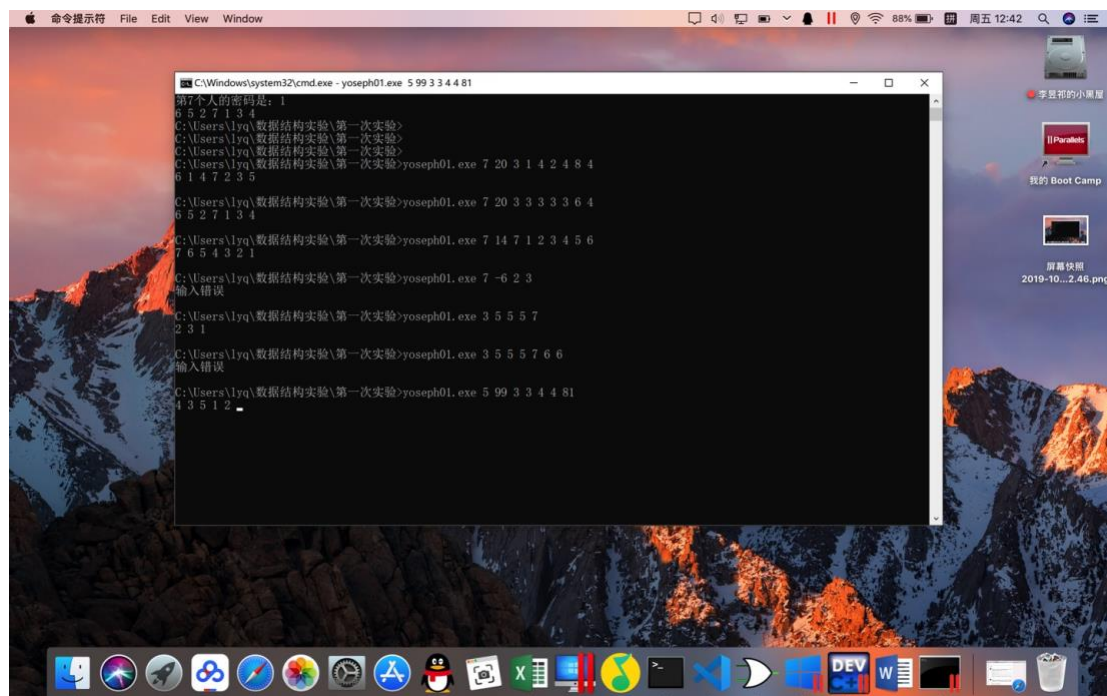
尝试在命令行中只输入文件名：



此时程序会继续执行，且提示用户继续输入参数！

2. 顺序表:

仿照链表进行了测试:



```
C:\Windows\system32\cmd.exe - yoseph01.exe 5 99 3 3 4 4 8 1
第7个人的密码是: 1
6 5 2 7 1 3 4
C:\Users\lyq\数据结构实验\第一次实验>
C:\Users\lyq\数据结构实验\第一次实验>
C:\Users\lyq\数据结构实验\第一次实验>yoseph01.exe 7 20 3 1 4 2 4 8 4
6 1 4 7 2 3 5
C:\Users\lyq\数据结构实验\第一次实验>yoseph01.exe 7 20 3 3 3 3 3 6 4
6 5 2 7 1 3 4
C:\Users\lyq\数据结构实验\第一次实验>yoseph01.exe 7 14 7 1 2 3 4 5 6
7 6 5 4 3 2 1
C:\Users\lyq\数据结构实验\第一次实验>yoseph01.exe 7 -6 2 3
输入错误
C:\Users\lyq\数据结构实验\第一次实验>yoseph01.exe 3 5 5 5 7
2 3 1
C:\Users\lyq\数据结构实验\第一次实验>yoseph01.exe 3 5 5 5 7 6 6
输入错误
C:\Users\lyq\数据结构实验\第一次实验>yoseph01.exe 5 99 3 3 4 4 8 1
4 3 5 1 2
```

六 . 实验总结

- 1.本学期第一次上机实验，对线性表章节进行了训练，着重练习了建表，遍历，删除，等基本操作。
- 2.熟悉了命令行传参的规则,以及带参数的 main 函数的编写。
- 3.理解线性表的顺序存储和链式存储的特性，掌握在不同存储结构，不同约定下，其基本操作的实现方法与差异。

七 . 附录

1. “yoseph00.c”（链表实现）
2. “yoseph01.c”（顺序表实现）