中国科学技术大学计算机学院
《数据结构》报告

实验题目：线性表的基础训练

学生姓名：王章瀚

学生学号：PB18111697

完成日期：2019 年 10 月 16 日

计算机实验教学中心制

2019 年 09 月

1 实验要求

1.1 概述

本次实验要求完成约瑟夫问题的求求解。约瑟夫问题的一中描述是：变好为 $1, 2, \ldots, n$ 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。要求设计一个程序求出出列顺序。

1.2 输入与输出

输入要求以命令行输入，格式为：

〈可执行程序名〉〈人数 n〉〈初始的报数上限 m〉〈密码 1〉……〈密码 n〉

其中，输入的人数 n，初始上限 m 及所有密码要求为正整数。

并且，当除可执行程序名外，没有参数时，将继续执行程序并提示用户输入这些参数

输出格式为：

〈第 1 个出列者序号〉〈第 2 个出列者序号〉 … 〈第 n 个出列者序号〉

1.3 测试数据

Input:
*.exe 7 20 3 1 7 2 4 8 4
Output:
6 1 4 7 2 3 5

2 设计思路

本实验的基本算法如下图所示：

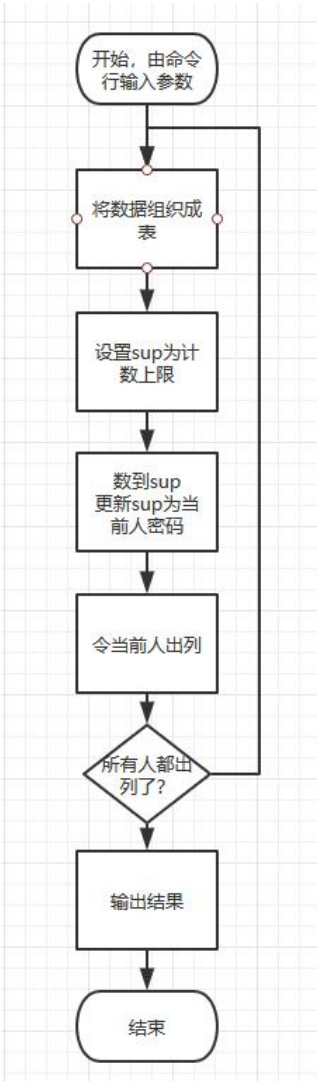

图 1: 基本算法思路

本实验中，使用两种数据结构来分别实现任务，即循环链表和线性表。下分类论之。

2.1 循环链表

2.1.1 数据类型的定义

有循环链表与循环链表的结点。

2.1.2 主程序流程

主程序读入命令行数据，若发现异常则向用户请求重新输入。此后调用 JosephRing 循环链表的 printAnswer() 来输出结果。

2.2 线性表

2.2.1 数据类型的定义

有线性表的节点，及用数组表示的线性表。

2.2.2 主程序流程

主程序读入命令行数据，若发现异常则向用户请求重新输入。此后调用 solve() 函数，将线性表作为参数传入来输出结果。

3 关键代码讲解

由于没有复杂的函数调用关系，故将函数调用关系图略去。

且此处算法设计思路极其简单，即不断往下数人，直到达到上限就让此人出列，如此循环，直至人均出列。

3.1 使用循环链表

3.1.1 数据类型的定义

1). 结点数据类型定义

```
1    class JosephNode {
2    public:
3        int index; // 结点序号
4        int code; //结点密码
5        JosephNode* next; // 下一个结点
6
```

```
        JosephNode(int index, int code) : index(index), code(code), next(
    nullptr) {};
    };

```

2). 循环链表数据类型定义

```
    class JosephRing {
    public:
        int num; // 总结点数
        int sup; //计数上限
        JosephNode* cur; //当前指向结点
        JosephNode* beforeCur; //当前指向结点的前一个结点
        JosephRing(int num, int sup, int* codes);
        void printAnswer();
    };

```

3.1.2 主要算法

主要算法即将流程图内容实现一遍，过程较为简单：

```
    void printAnswer() {
        FILE* f = fopen("answer.txt", "w");
        while (num--) {
            sup--;
            //不断数下去，直到数完了上限
            while (sup--) {
                beforeCur = cur;
                cur = cur->next;
            }
            cout << cur->index << ' ';
            fprintf(f, "%d ", cur->index);
            sup = cur->code;
            beforeCur->next = cur->next;
            JosephNode* toDelete = cur;
            cur = cur->next;
            delete toDelete; //删除去掉了的节点占用的内存空间
        }
        cout << endl;
        fprintf(f, "\n");
    }
```

```

```

3.2 使用线性表

3.2.1 数据类型的定义

1). 结点数据类型定义

```
class JosephNode {
public:
    int index; // 结点序号
    int code; // 结点密码
    int nextCur; // 下一个结点的位置
    JosephNode(int index, int code, int nextCur)
    : index(index), code(code), nextCur(nextCur) {};
};

```

3.2.2 主要算法

主要算法即将流程图内容实现一遍，过程较为简单：

```
void solve(int num, int m, JosephNode** list) {
    FILE* f = fopen("answer.txt", "w");
    int beforeCur = num - 1;
    int cur = 0;
    while (num--) {
        m--;
        // 循环数人，直到达到m
        while (m--) {
            beforeCur = cur;
            cur = list[cur]->nextCur;
        }
        cout << list[cur]->index << ' ';
        fprintf(f, "%d ", list[cur]->index);
        m = list[cur]->code;
        list[beforeCur]->nextCur = list[cur]->nextCur;
        cur = list[cur]->nextCur;
    }
    cout << endl;
    fprintf(f, "\n");
```

```
    }

```

3). 关于将命令行读入字符串转化为 int 型数据的代码

此处使用了 sscanf 函数来实现

```
//从字符串中读取一个正整数
bool tryReadNum(int* dest, char* s) {
    int len = strlen(s);
    for (int i = 0; i < len; i++) {
        if (!isdigit(s[i])) {
            cout << "出现非正整数\n";
            return false;
        }
    }
    sscanf(s, "%d", dest);
}

```

4 调试分析

4.1 问题发现与解决

调试过程中发现设计的循环链表并不能做到循环的效果，而成了单向链表。经 Debug 模式逐步检验，发现最后未将头结点作为尾结点的下一节点，因而改正。

4.2 算法的时空复杂度分析

假设输入的数据为

$$*.exe\ n\ m_0\ m_1\ \ldots\ m_n$$

以下分析对循环链表和线性表均有效：

由于算法是一步一步寻找到下一个人来执行的，故显然其时间复杂度为 $o(\sum_{i=0}^{n} m_i)$

此外，空间复杂度上，只需要存储这 n 个人的密码，序号等信息，故空间复杂度是显然的

5 代码测试

5.1 data1

Input:
约瑟夫环.exe 7 20 3 1 7 2 4 8 4
Output:
6 1 4 7 2 3 5

截图如下：

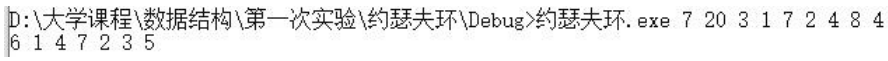

图 2: data1

5.2 data2

Input:
约瑟夫环.exe 10 50 16 48 35 94 1 6 4 1 9 64
Output:
10 1 9 3 8 2 7 4 6 5

截图如下：

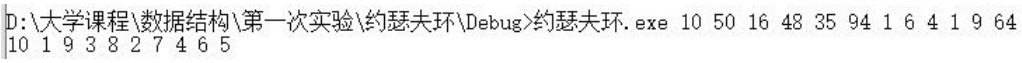

图 3: data2

6 实验总结

通过本次的实验，本人更加熟悉了循环链表和线性表的构造和使用，能够更加灵活地运用这两种数据结构来解决问题。此外，还复习了命令行参数的使用，加深了印象。

7 附录

7.1 约瑟夫问题（循环链表实现）完整代码

```cpp
#include <iostream>
#include <cstdio>
#include <cstring>

using namespace std;

// 约瑟夫循环链表结点
class JosephNode {
public:
    int index; //结点序号
    int code; //结点密码
    JosephNode* next; //下一个结点

    JosephNode(int index, int code) : index(index), code(code), next(nullptr) {};
};

// 约瑟夫循环链表
class JosephRing {
public:
    int num; // 总结点数
    int sup; // 计数上限
    JosephNode* cur; //当前结点
    JosephNode* beforeCur; //当前结点的前一个结点
    JosephRing(int num, int sup, int* codes) : num(num), sup(sup){
        cur = new JosephNode(1, codes[0]);
        JosephNode* p = cur;
        for (int i = 1; i < num; i++) {
            p->next = new JosephNode(i + 1, codes[i]);
            p = p->next;
        }
        beforeCur = p;
        p->next = cur;
    }

    // 输出结果
    void printAnswer() {
        FILE* f = fopen("answer.txt", "w");
        while (num--) {
            sup--;
            //不断数下去，直到数完了上限
            while (sup--) {
                beforeCur = cur;
                cur = cur->next;
            }
            cout << cur->index << ' ';
```

```
46                  fprintf(f, "%d ", cur->index);
47                  sup = cur->code;
48                  beforeCur->next = cur->next;
49                  JosephNode* toDelete = cur;
50                  cur = cur->next;
51                  delete toDelete; //删除去掉了的节点占用的内存空间
52              }
53              cout << endl;
54              fprintf(f, "\n");
55          }
56      };
57
58      //从字符串中读取一个正整数
59      bool tryReadNum(int* dest, char* s) {
60          int len = strlen(s);
61          for (int i = 0; i < len; i++) {
62              if (!isdigit(s[i])) {
63                  cout << "出现非正整数\n";
64                  return false;
65              }
66          }
67          sscanf(s, "%d", dest);
68      }
69
70      int main(int argc, char* argv[])
71      {
72          bool flag = true;
73          // 对命令行参数的数量做分类，要满足题目要求，至少有4个参数
74          if (argc >= 4) {
75              int* codes = new int[argc - 1];
76              // 将命令行参数转化为int类型数据存储在code中
77              for (int i = 1; i < argc; i++) {
78                  if (!tryReadNum(codes + i - 1, argv[i])) {
79                      flag = false;
80                      break;
81                  }
82              }
83              // 判断人数和密码数是否对应
84              if (codes[0] != argc - 3 && flag) {
85                  cout << "人数或密码数有误\n";
86                  flag = false;
87              }
88              if (flag) {
89                  // 实例化JosephRing,调用其printAnswer()方法以完成题目
90                  JosephRing* jr = new JosephRing(codes[0], codes[1], codes +
    2);
```

```
91                  jr->printAnswer();
92              }
93          }
94          if(!flag) {
95              // 命令行参数格式不正确，要求重新输入。
96              int num, m;
97              cout << "命令行信息不完整，请重新输入，按照如下格式：\n"
98              << "〈人数n〉〈初始的报数上限m〉〈密码1〉 …… 〈密码n〉\n" <<
    endl;
99              cin >> num >> m;
100             int* codes = new int[num];
101             for (int i = 0; i < num; i++) {
102                 cin >> codes[i];
103             }
104             // 实例化JosephRing,调用其printAnswer()方法以完成题目
105             JosephRing* jr = new JosephRing(num, m, codes);
106             jr->printAnswer();
107         }
108     }
109
```

7.2 约瑟夫问题（线性宝实现）完整代码

```
1       #include <iostream>
2       #include <cstdio>
3       #include <cstring>
4
5       using namespace std;
6
7       // 约瑟夫结点
8       class JosephNode {
9       public:
10          int index; //结点序号
11          int code; //结点密码
12          int nextCur; //下一个结点的位置
13          JosephNode(int index, int code, int nextCur)
14          : index(index), code(code), nextCur(nextCur) {};
15      };
16
17      // 从字符串s中读取正整数，存入dest中
18      bool tryReadNum(int* dest, char* s) {
19          int len = strlen(s);
20          for (int i = 0; i < len; i++) {
```

```
            if (!isdigit(s[i])) {
                cout << "出现非正整数\n";
                return false;
            }
        }
    sscanf(s, "%d", dest);
    return true;
    }

    // 解约瑟夫问题
    void solve(int num, int m, JosephNode** list) {
        FILE* f = fopen("answer.txt", "w");
        int beforeCur = num - 1;
        int cur = 0;
        while (num--) {
            m--;
            // 循环数人，直到达到m
            while (m--) {
                beforeCur = cur;
                cur = list[cur]->nextCur;
            }
            cout << list[cur]->index << ' ';
            fprintf(f, "%d ", list[cur]->index);
            m = list[cur]->code;
            list[beforeCur]->nextCur = list[cur]->nextCur;
            cur = list[cur]->nextCur;
        }
        cout << endl;
        fprintf(f, "\n");
    }

    int main(int argc, char* argv[])
    {
        int num, m, code;
        JosephNode** list = nullptr;
        bool flag = true;
        // 对命令行参数的数量做分类，要满足题目要求，至少有4个参数
        if (argc >= 4) {
            // 读取人数
            if (!tryReadNum(&num, argv[1])) {
                flag = false;
            }
            // 判断人数和密码数是否一致
            if (num != argc - 3 && flag) {
                cout << "人数或密码数有误\n";
                flag = false;
```

```
67                }
68                // 读取初始密码
69                if (!tryReadNum(&m, argv[2]) && flag) {
70                    flag = false;
71                }
72                if (flag) {
73                    list = new JosephNode*[num];
74                    // 读取所有密码
75                    for (int i = 0; i < num; i++) {
76                        if (!tryReadNum(&code, argv[i + 3])) {
77                            flag = false;
78                            break;
79                        }
80                        list[i] = new JosephNode(i + 1, code, i == num - 1 ?
      0 : i + 1);
81                    }
82                    // 执行
83                    solve(num, m, list);
84                    return 0;
85                }
86            }
87            if (flag) {
88                // 命令行输入信息不完整
89                cout << "命令行信息不完整，请重新输入，按照如下格式：\n"
90                << "〈人数n〉 〈初始的报数上限m〉 〈密码1〉 …… 〈密码n〉\n"
91                << endl;
92                // 重新读取人数和初始密码
93                cin >> num >> m;
94                list = new JosephNode*[num];
95                // 重新读取密码
96                for (int i = 0; i < num; i++) {
97                    cin >> code;
98                    list[i] = new JosephNode(i + 1, code, i == num - 1 ? 0 :
      i + 1);
99                }
100               // 执行
101               solve(num, m, list);
102           }
103       }
104
105
```