项目说明文档

数据结构课程设计
——约瑟夫生者死者游戏

作 者 姓 名：杨鑫

学　　号：1950787

指 导 教 师：张颖

学院、 专业：软件学院 软件工程

同济大学

Tongji University

目录

1 分析.....1
1.1 背景分析.....1
1.2 功能分析.....1
2 设计.....2
2.1 数据结构设计.....2
2.2 类结构设计.....2
2.3 成员与操作设计.....2
2.4 系统设计.....3
3 实现.....4
3.1 初始化游戏.....4
3.1.1 初始化游戏流程图.....4
3.1.2 初始化游戏核心代码.....4
3.1.3 初始化游戏截屏示例.....5
3.2 游戏主体功能.....6
3.2.1 游戏主体功能流程图.....6
3.1.2 游戏主体功能核心代码.....6
3.2.3 游戏主体功能截屏示例.....7
4 测试.....7
4.1 整体功能测试.....7
4.2 边界测试.....7
4.2.1 当输入数据均为最小值时.....7
4.3 出错测试.....8
4.3.1 人数等输入不合法.....8

1 分析

1.1 背景分析

约瑟夫生者死者游戏的大意是：30 个旅客同乘一条船，因为严重超载，加上风高浪大危险万分；因此船长告诉乘客，只有将全船一半的旅客投入海中，其余人才能幸免于难。无奈，大家只得统一这种方法，并议定 30 个人围成一圈，由第一个人开始，依次报数，数到第 9 人，便将他投入大海中，然后从他的下一个人数起，数到第 9 人，再将他投入大海，如此循环，直到剩下 15 个乘客为止。现在需要一个程序，在输入相应的规则后能够准确的计算出每次被扔出去的乘客以及最后的幸存者。

1.2 功能分析

本游戏的数学建模如下：假如 N 个旅客排成一个环形，依次顺序编号 1，2，…，N。从某个指定的第 S 号开始。沿环计数，每数到第 M 个人就让器出列，且从下一个人开始重新计数，继续进行下去。这个过程一直进行到剩下 K 个旅客为止。

本游戏要求用户输入的内容包括：

1、 旅客的个数，也就是 N 的值；

2、 离开旅客的间隔书，也就是 M 的值；

3、 所有旅客的序号作为一组数据要求存放在某种数据结构中。

本游戏要求输出的内容是包括：

1. 离开旅客的序号；

2. 剩余旅客的序号。

2 设计

2.1 数据结构设计

按照上述要求，本程序采用了循环链表的数据结构，并且没有附加头结点。这样子可以比较便利的寻找到每次的结果。循环链表有一个 head 指针和 current 指针元素，这样便于记录起始位置和当前位置，使得编写代码更加方便。

2.2 类结构设计

设计循环链表需要两个类：一个是节点类（ListNode 类）和循环链表（CircleList 类）。前者保存了相关的数据和指向下一个节点的指针，后者集合了所有的节点，并实现了按规则删除，打印等操作。在此处，本程序将 ListNode 设计为了结构体，这样操作更加方便灵活。本程序还有一个系统类（System 类），封装了交互操作和对应游戏规则的执行等功能。

2.3 成员与操作设计

循环链表结点结构体（ListNode）：

成员：

```
int data; // 节点数据
ListNode * link; // 后驱指针
```

函数：

```
ListNode() : data(0), link(NULL) {} // 默认构造函数
ListNode(int newData, ListNode* newLink = NULL) : data(newData), link(newLink) {} // 带参数的工作函数
ListNode(const ListNode& N) : data(N.data), link(N.link) {} // 拷贝构造函数
~ListNode() {} // 析构函数
```

循环链表类（CircleList）：

私有成员：

```
ListNode* head, * current; // 头指针和当前位置指针
int deadnum; // 死亡数字
```

公有操作：

```
CircleList() : head(NULL), current(NULL), deadnum(0) {} // 默认构造函数
CircleList(int num); // 构造函数
~CircleList(); // 析构函数
void SetStart(int start); // 设置起始位置
void SetDead(int dead) { deadnum = dead; } // 设置死亡数字
```

```
    int Dead(); // 执行一次死亡操作
    void Print(); // 打印
```

系统类（System）：

私有成员：

```
    int livenums; // 幸存者数量
    CircleList* circle; // 循环链表，存放信息
```

公有操作：

```
    System() : livenums(0) { circle = new CircleList; } // 构造函数
    ~System() { delete circle; } // 析构函数
    void loop(); // 主循环
    void PrintLives(); // 打印幸存者
```

2.4 系统设计

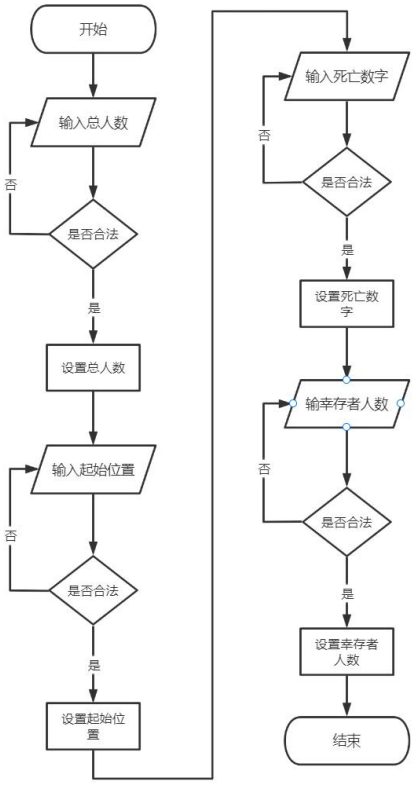
在实例化一个系统类之后进入其主循环函数，提示用户输入一些需要的数据，然后按照规则执行游戏之后，将每次死亡的人和最终的幸存者打印出来，从而完成游戏的完整执行。

程序兼容了 windows 和 LINUX 平台，在双平台下均可以正常运行。

3 实现

3.1 初始化游戏

3.1.1 初始化游戏流程图



3.1.2 初始化游戏核心代码

```
1.      // 总人数必须大于等于 2
2.      while (totNums < 2) {
3.          // 清空缓冲区
4.          cin.clear();
5.          cin.ignore(numeric_limits<streamsize>::max(), '\n');
6.          cout << "总人数必须为大于 1 的正整数！" << endl;
7.          cout << "请重新输入：  ";
8.          cin >> totNums;
9.      }
10.     circle = new CircleList(totNums);
11.
12.     // 检验起始位置是否合法
13.     while (startNum < 1 || startNum > totNums) {
```

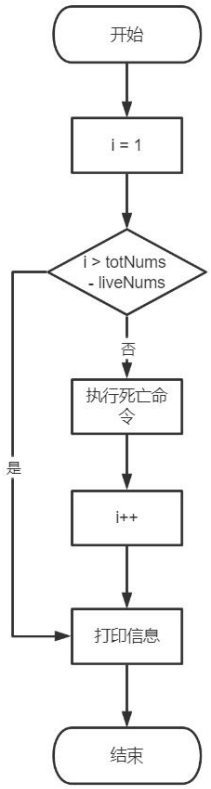
```
14.             cin.clear();
15.             cin.ignore(numeric_limits<streamsize>::max(), '\n');
16.             cout << "开始位置必须大于等于 1 且小于总人数！" << endl;
17.             cout << "请重新输入：  ";
18.             cin >> startNum;
19.         }
20.         // 设置起始位置
21.         circle->SetStart(startNum);
22.
23.         // 检验死亡数字是否合法
24.         while (deadNum <= 1) {
25.             cin.clear();
26.             cin.ignore(numeric_limits<streamsize>::max(), '\n');
27.             cout << "死亡数字必须为大于 1 的正整数！" << endl;
28.             cout << "请重新输入：  ";
29.             cin >> deadNum;
30.         }
31.         // 设置死亡数字
32.         circle->SetDead(deadNum);
33.
34.         // 检验幸存者数量是否合法
35.         while (liveNums < 1 || liveNums >= totNums) {
36.             cin.clear();
37.             cin.ignore(numeric_limits<streamsize>::max(), '\n');
38.             cout << "剩余人数必须大于等于 1 且小于总人数！" << endl;
39.             cout << "请重新输入：  ";
40.             cin >> liveNums;
41.         }
42.         livenums = liveNums;
```

3.1.3 初始化游戏截屏示例

```
输入 S 以开始约瑟夫生死环游戏，输入 E 或其他以退出游戏.....
S
游戏规则：现有N人围成一圈，从第S个人开始一次报数，报M的人出局，再由下一人开始报数，如此循环，直至剩下K个人为止。

请输入生死游戏的总人数N：12

请输入游戏开始的位置S：3

请输入死亡数字M：7

请输入剩余的生者人数K：3
```

3.2 游戏主体功能

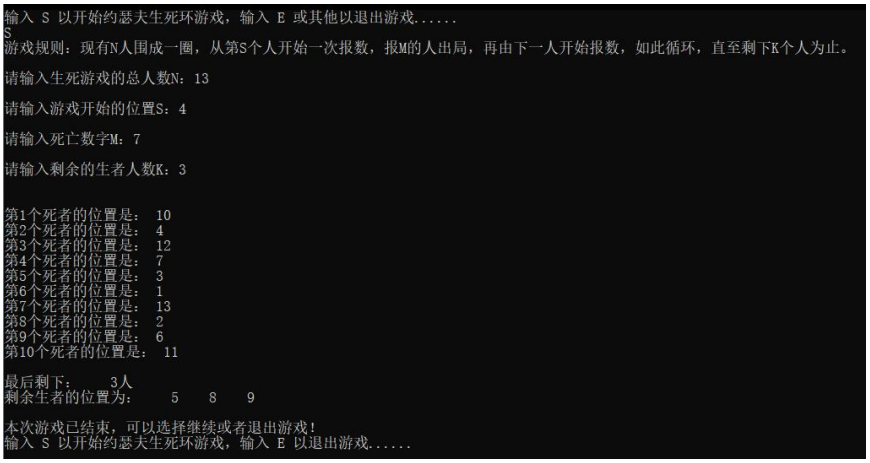
3.2.1 游戏主体功能流程图



3.1.2 游戏主体功能核心代码

```
  // 执行 totNums - liveNums 次死亡操作
  for (int i = 1; i <= totNums - liveNums; i++) {
      int die = circle->Dead();
      cout << "第" << i << "个死者的位置是:  " << die << endl;
  }
cout << endl;

// 打印幸存者信息
PrintLives();
```

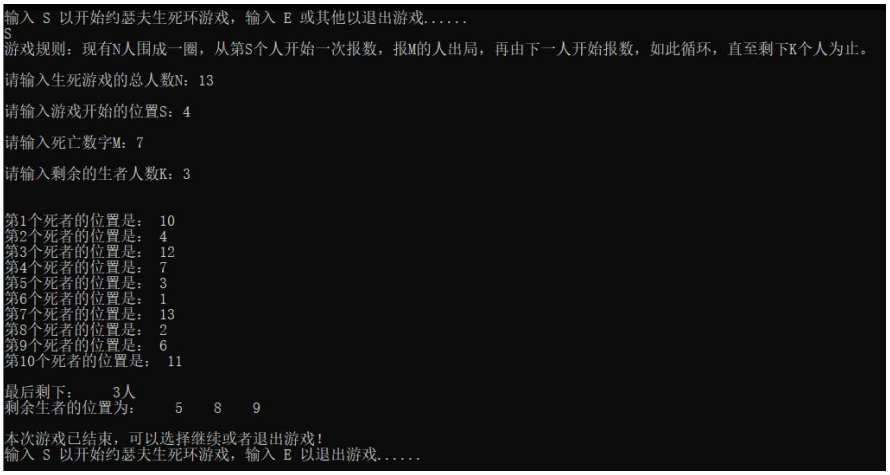
3.2.3 游戏主体功能截屏示例



4 测试

4.1 整体功能测试

运行结果截图：



4.2 边界测试

4.2.1 当输入数据均为最小值时

处理方式：系统正常运行，程序不会异常退出或者崩溃。
运行结果截图：

```
输入 S 以开始约瑟夫生死环游戏，输入 E 或其他以退出游戏.....
S
游戏规则：现有N人围成一圈，从第S个人开始一次报数，报M的人出局，再由下一人开始报数，如此循环，直至剩下K个人为止。

请输入生死游戏的总人数N：13

请输入游戏开始的位置S：1

请输入死亡数字M：2

请输入剩余的生者人数K：1


第1个死者的位置是： 2
第2个死者的位置是： 4
第3个死者的位置是： 6
第4个死者的位置是： 8
第5个死者的位置是： 10
第6个死者的位置是： 12
第7个死者的位置是： 1
第8个死者的位置是： 5
第9个死者的位置是： 9
第10个死者的位置是： 13
第11个死者的位置是： 7
第12个死者的位置是： 3

最后剩下：     1人
剩余生者的位置为：     11

本次游戏已结束，可以选择继续或者退出游戏！
输入 S 以开始约瑟夫生死环游戏，输入 E 以退出游戏.....
```

4.3 出错测试

4.3.1 人数等输入不合法

处理方式：系统正常运行，程序不会异常退出或者崩溃。

运行结果截图：

```
输入 S 以开始约瑟夫生死环游戏，输入 E 或其他以退出游戏.....
S
游戏规则：现有N人围成一圈，从第S个人开始一次报数，报M的人出局，再由下一人开始报数，如此循环，直至剩下K个人为止。

请输入生死游戏的总人数N：0
总人数必须为大于 1 的正整数！
请重新输入： -1
总人数必须为大于 1 的正整数！
请重新输入：
```

```
输入 S 以开始约瑟夫生死环游戏，输入 E 或其他以退出游戏.....
S
游戏规则：现有N人围成一圈，从第S个人开始一次报数，报M的人出局，再由下一人开始报数，如此循环，直至剩下K个人为止。

请输入生死游戏的总人数N：12

请输入游戏开始的位置S：3

请输入死亡数字M：0
死亡数字必须为大于 1 的正整数！
请重新输入：
```