**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： 戴仁杰

学 号： 1951650

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 - 3 -](#_Toc90988644)

[1.1 背景分析 - 3 -](#_Toc90988645)

[1.2 功能分析 - 3 -](#_Toc90988646)

[2 设计 - 4 -](#_Toc90988647)

[2.1 数据结构设计 - 4 -](#_Toc90988648)

[2.2 类结构设计 - 4 -](#_Toc90988649)

[2.3 成员与操作设计 - 4 -](#_Toc90988650)

[2.3.1 Node类： - 4 -](#_Toc90988651)

[2.3.2 LinkList类： - 4 -](#_Toc90988652)

[2.3.3 JosephusGame类 - 5 -](#_Toc90988653)

[2.4 系统设计 - 5 -](#_Toc90988654)

[3 实现 - 6 -](#_Toc90988655)

[3.1 建立玩家系统功能的实现 - 6 -](#_Toc90988656)

[3.1.1 建立玩家系统功能流程图 - 6 -](#_Toc90988657)

[3.1.2 建立玩家系统功能核心代码 - 6 -](#_Toc90988658)

[3.2 删除死亡玩家功能的实现 - 7 -](#_Toc90988659)

[3.2.1 删除死亡玩家功能流程图 - 7 -](#_Toc90988660)

[3.2.2 删除死亡玩家功能核心代码 - 7 -](#_Toc90988661)

[3.3 展示最后存活玩家功能的实现 - 8 -](#_Toc90988662)

[3.3.1 展示最后存活玩家功能流程图 - 8 -](#_Toc90988663)

[3.3.2 展示最后存活玩家功能核心代码 - 8 -](#_Toc90988664)

[3.4 总体功能的实现 - 9 -](#_Toc90988665)

[3.4.1 总体功能流程图 - 9 -](#_Toc90988666)

[3.4.2 总体功能截屏示例 - 9 -](#_Toc90988667)

[4 测试 - 10 -](#_Toc90988668)

[4.1 一般功能测试 - 10 -](#_Toc90988669)

[4.2 边界测试 - 10 -](#_Toc90988670)

[4.2.1 死亡数字为1（报数即死亡）时 - 10 -](#_Toc90988671)

[4.2.2 死亡数字为N（第一次即删除尾结点）时 - 10 -](#_Toc90988672)

[4.3 出错测试 - 11 -](#_Toc90988673)

[4.3.1 输入为负数 - 11 -](#_Toc90988674)

[4.3.2 输入为字母 - 11 -](#_Toc90988675)

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。问哪些位置是将被扔下大海的位置。

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。

由于本游戏是玩家围成一个环形循环读数选择抛弃，所以旅客的序号可以抽象成为一个循环链表，并记录链表长度。输入旅客个数后，建立循环链表，然后从某个位置开始计数，到达指定位置后删除结点，然后重新计数，循环操作直至链表中剩余结点数目符合要求。

综上所述，约瑟夫生者死者游戏程序需要有输入、计数、删除等功能。

2 设计

2.1 数据结构设计

如上功能分析所述，该系统使用了循环链表数据结构，由于玩家报数时只需要朝一个方向进行，因此最后选择使用单向循环链表来存储信息。同时在链表前附加了一个头结点，这样使得增加和删除头结点时与处理其它结点方法相同，简化了代码。

2.2 类结构设计

链表一般包括两个抽象数据类型——链表结点类（Node类）与链表类（Linklist类）。程序中链表为单向循环链表，Node类来储存每一位玩家信息，Linklist类来将玩家信息整合在一起，并提供查询，插入，删除等操作。除此之外，本系统还设计了JosephusGame类，给用户提供与系统交互的接口。为了增强类的封装性，在数据交互时提供了一系列的get和set函数。同时为了使链表更加具有泛用性，本系统将Node类与Linklist类都设计为了模板类。

2.3 成员与操作设计

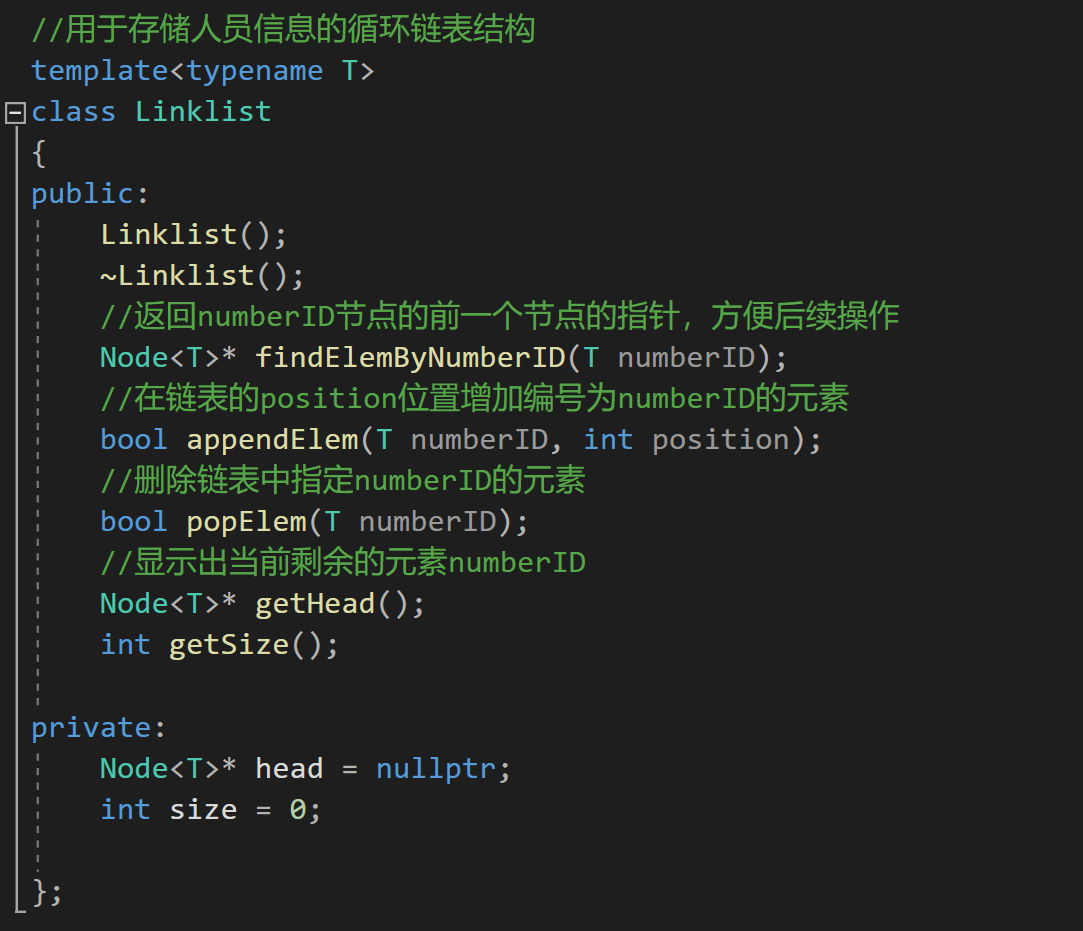
### 2.3.1 Node类：

Node类为链表节点类，采用了类模板提高泛用性。提供默认构造函数、带参构造函数以及与外接数据交流的一系列get与set函数。类内数据为data数据域以及指针域，存储玩家编号以及与后继节点相连。



### 2.3.2 LinkList类：

Linklist类为链表类，采用了类模板提高泛用性。提供构造函数新建链表，析构函数释放内存。提供getLinkSize()函数获得链表大小，并提供增删查改的各类函数供外界调用。类内数据为表头指针和链表大小。



### 2.3.3 JosephusGame类

JosephusGame类为约瑟夫生死者游戏类。对链表类（Linklist类）进行封装，丰富了交互提示信息，增强了程序输入错误、边界错误处理性能。类内数据为一个存储学生信息的链表，和游戏初始所需要的人数、起始位置、死亡数字、存活人数。



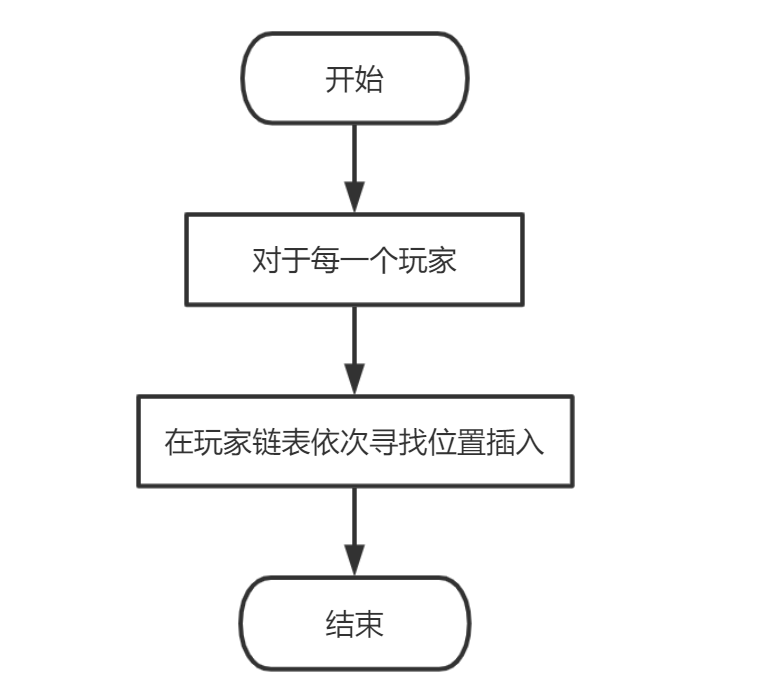
2.4 系统设计

系统首先会询问玩家输入生死游戏的总人数、游戏开始的位置、死亡数字、剩余生者人数，之后程序根据上述信息建立链表和游戏系统。之后调用JosephusGame类的killPerson()函数删除玩家并输出死者位置，然后调用showLivers()函数展示最后存活的玩家位置。

3 实现

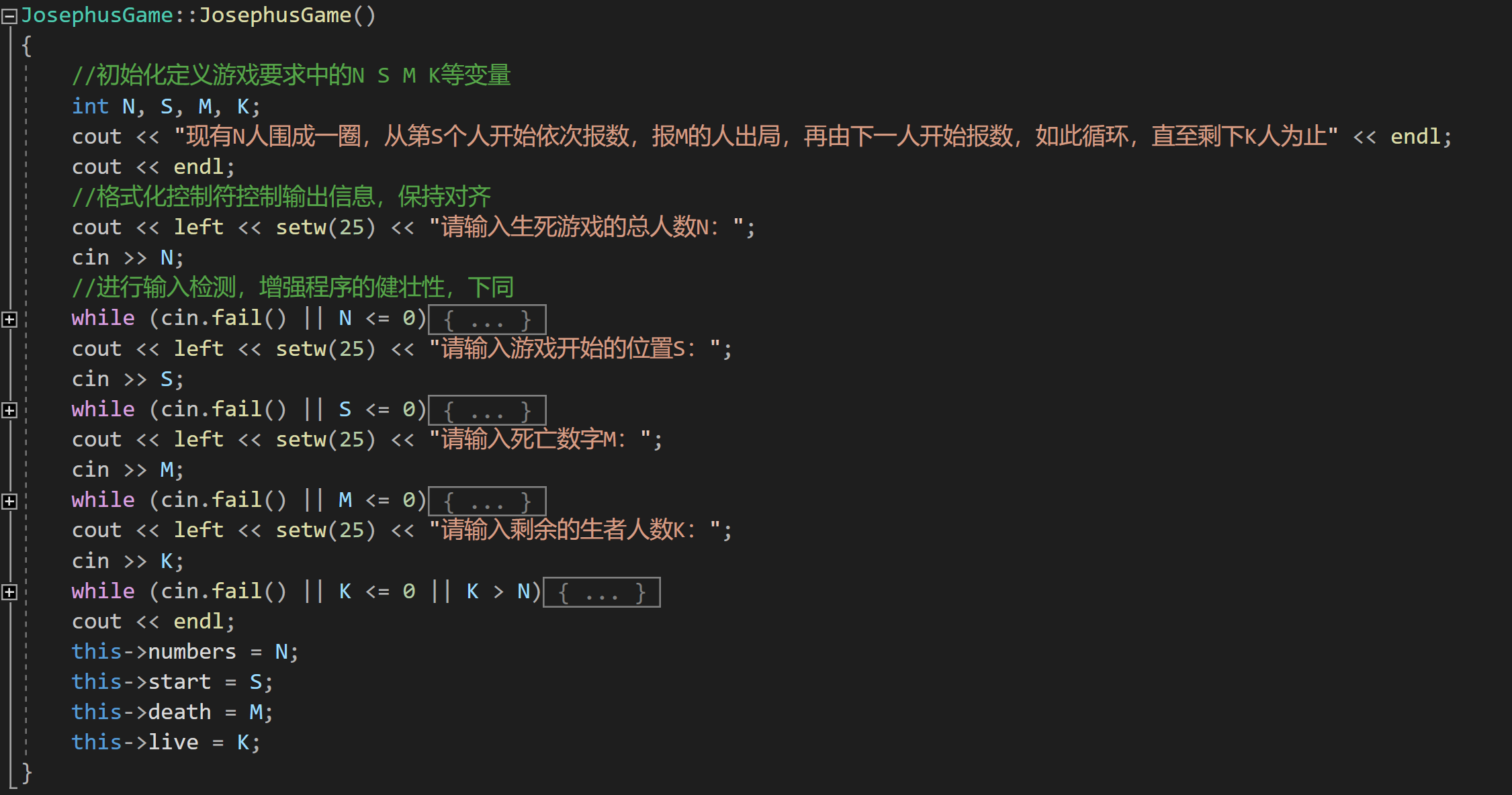
3.1 建立玩家系统功能的实现

3.1.1 建立玩家系统功能流程图

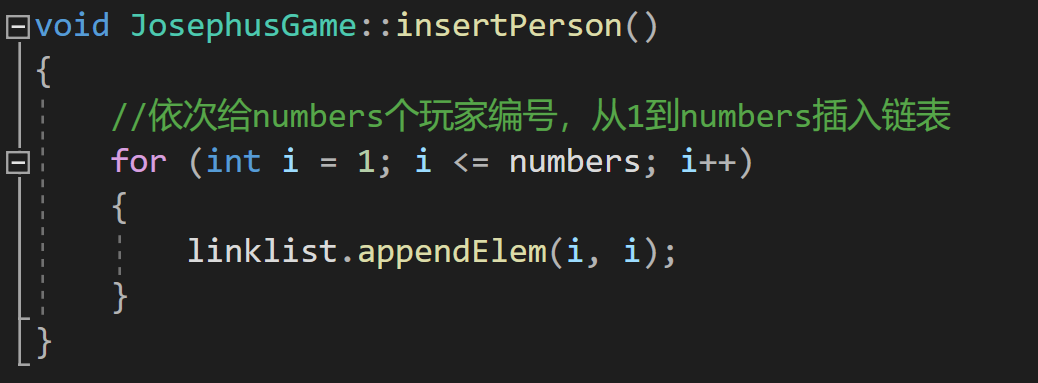


3.1.2 建立玩家系统功能核心代码

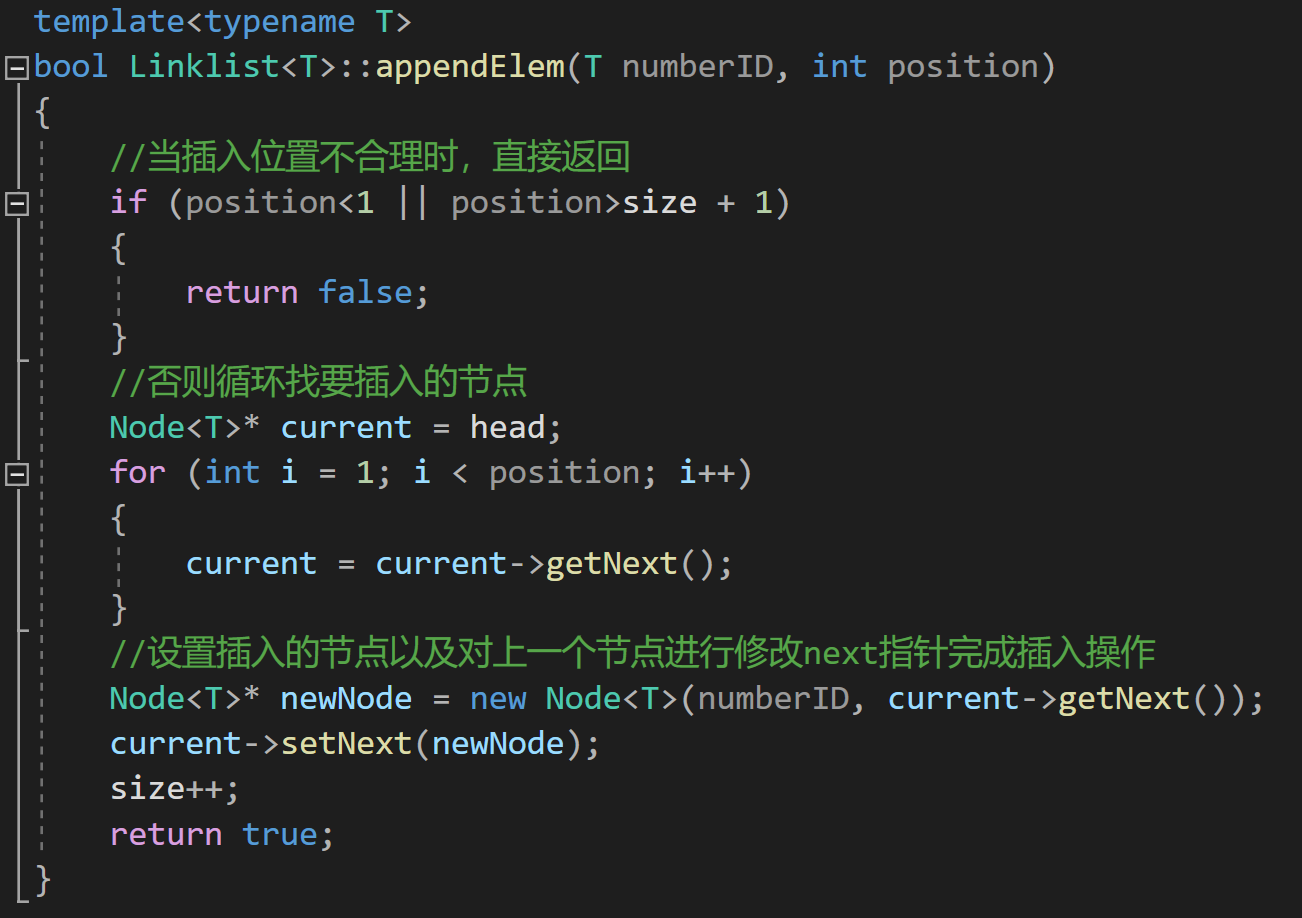
1. JosephusGame类的构造函数



2. JosephusGame类的insertPerson()函数

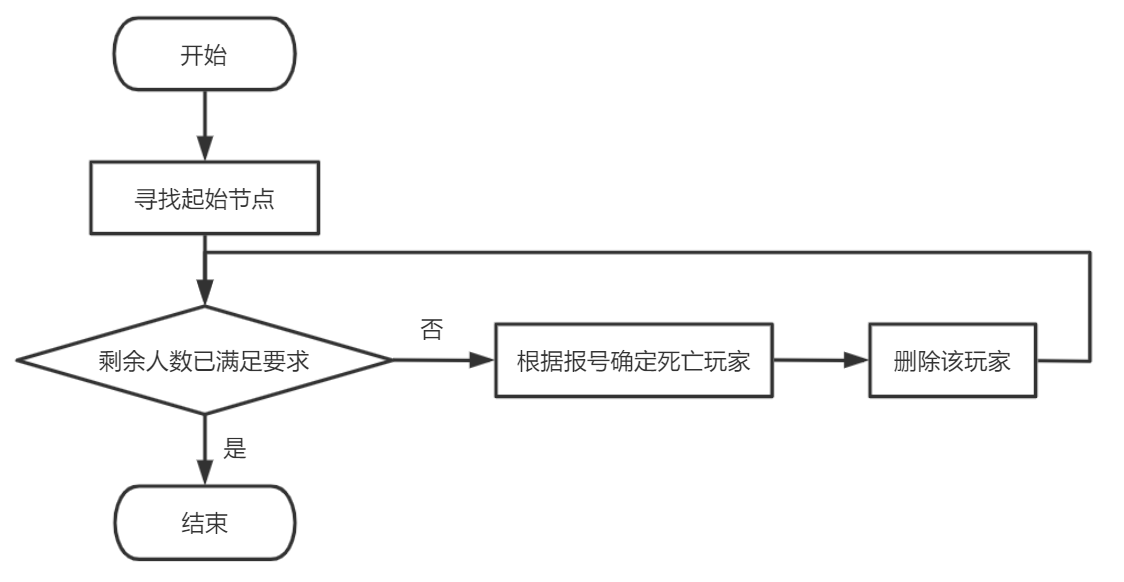


3.Linklist类的appendElem(T numberID, int position)函数



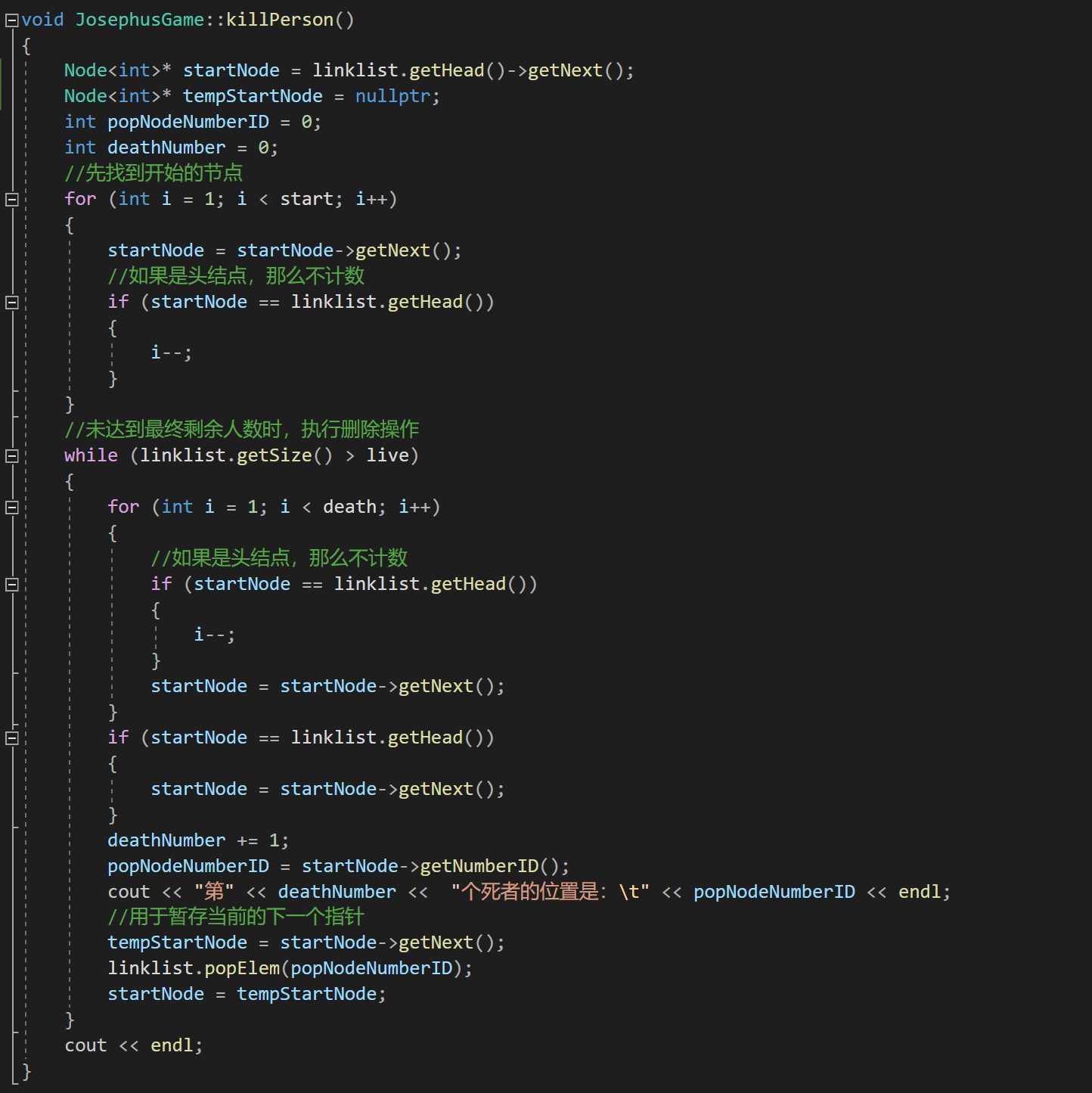
## 3.2 删除死亡玩家功能的实现

### 3.2.1 删除死亡玩家功能流程图

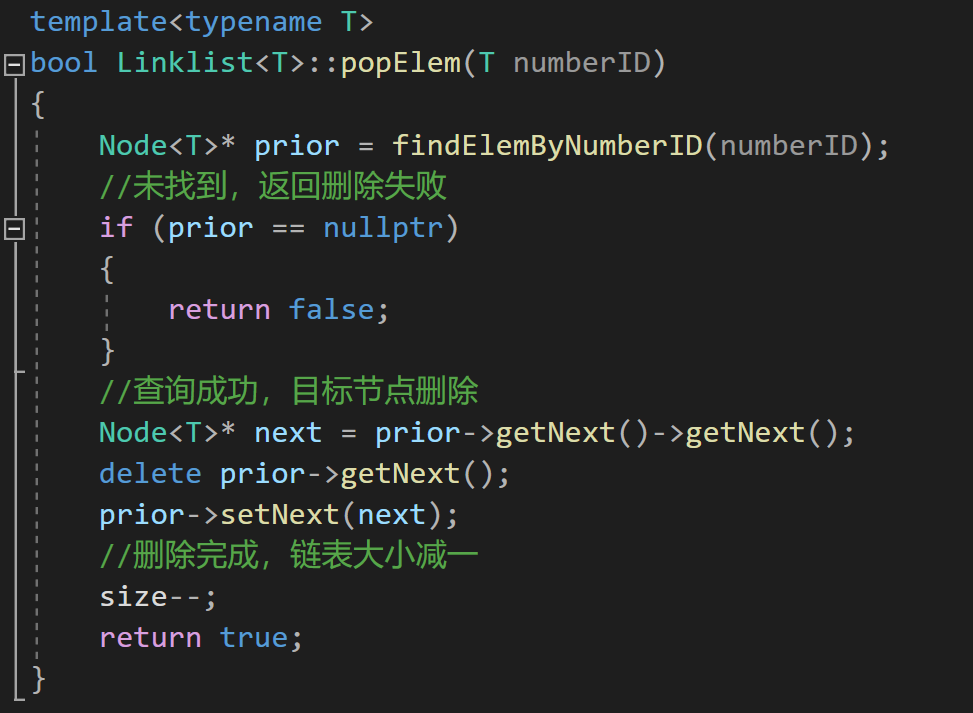


### 3.2.2 删除死亡玩家功能核心代码

1. JosephusGame类的killPerson()函数

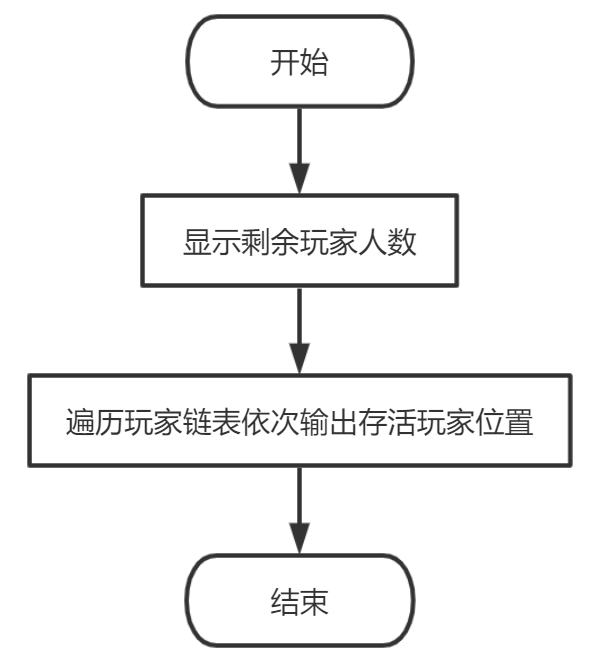


2. Linklist类的popElem(T numberID)函数



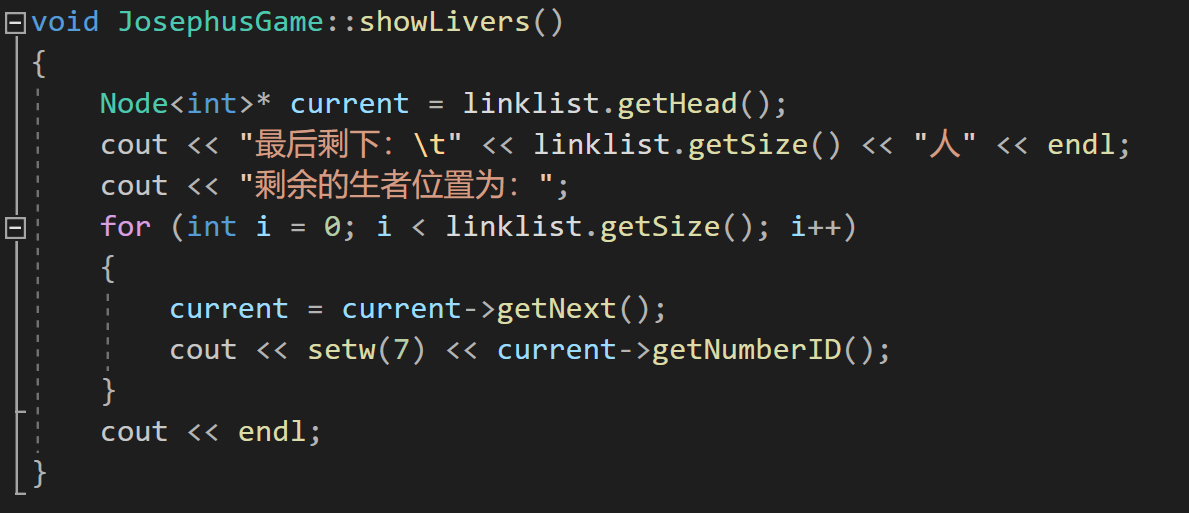
3.3 展示最后存活玩家功能的实现

3.3.1 展示最后存活玩家功能流程图



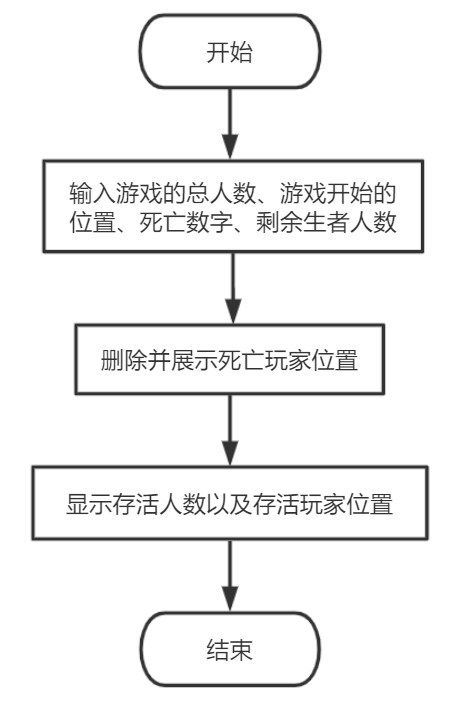
3.3.2 展示最后存活玩家功能核心代码

1. JosephusGame类中的showLivers()函数

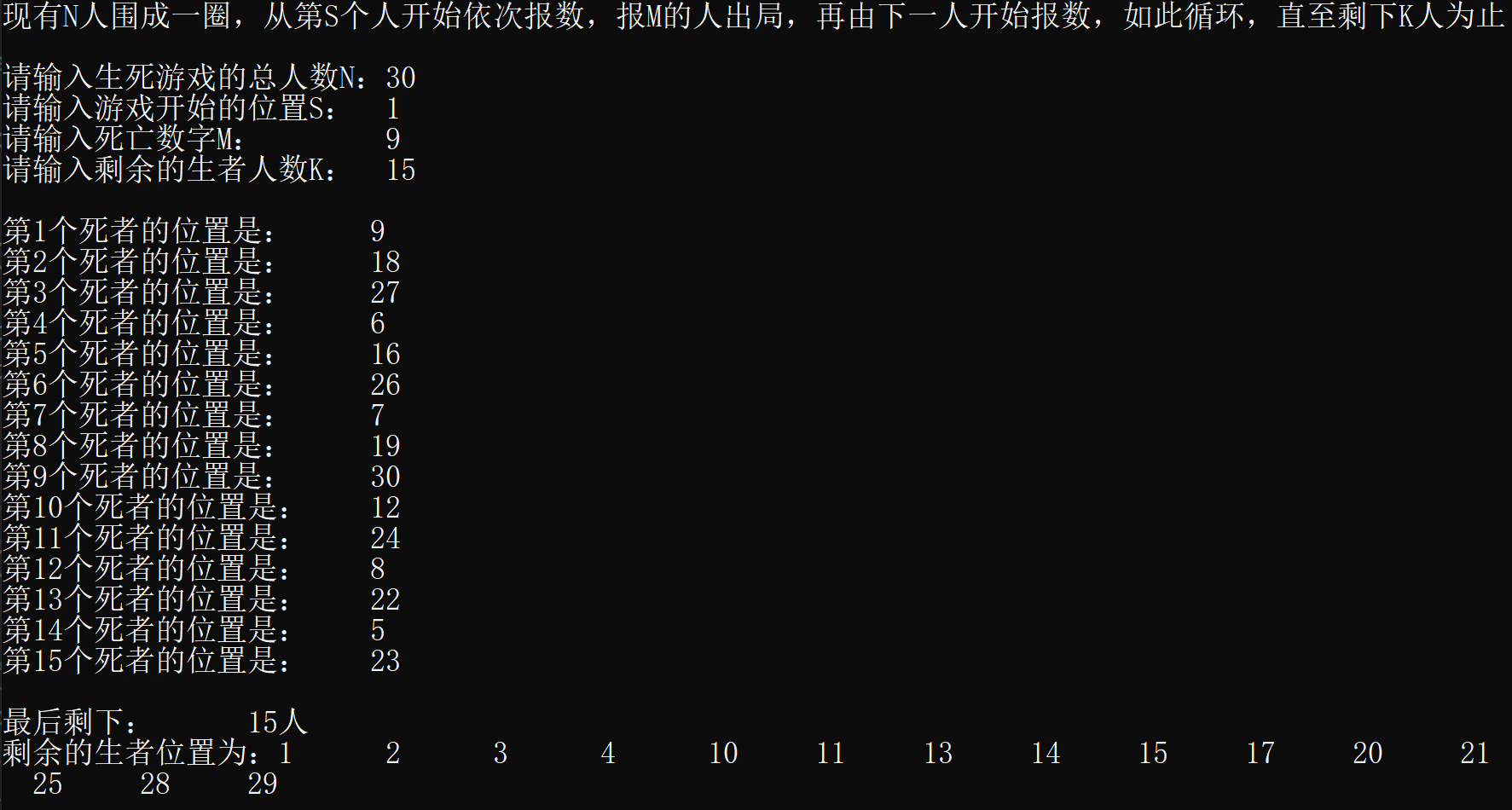


## 3.4 总体功能的实现

### 3.4.1 总体功能流程图



### 3.4.2 总体功能截屏示例

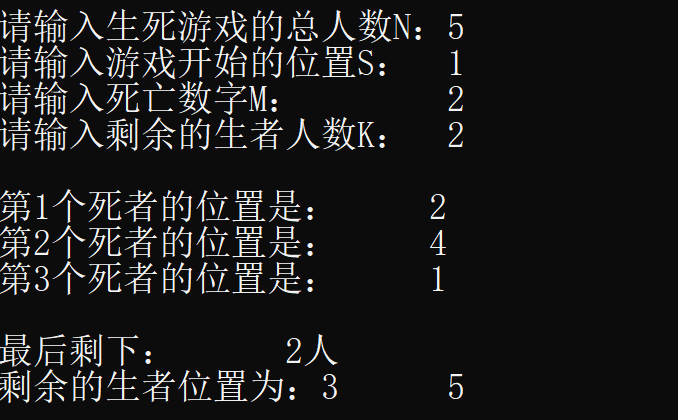


4 测试

4.1 一般功能测试

**测试用例**：总人数N=5，开始位置S=1，死亡数字M=2，剩余人数K=2

**实验结果：符合预期（截图如下）**

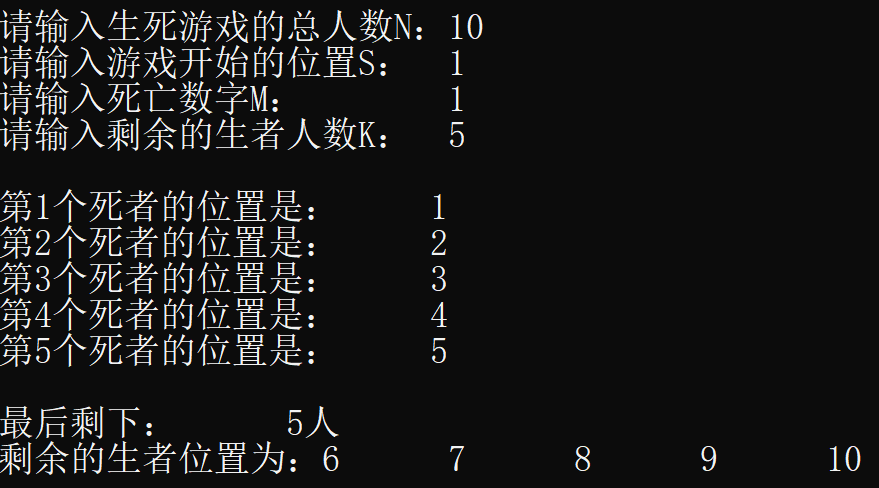


4.2 边界测试

4.2.1 死亡数字为1（报数即死亡）时

**测试用例：**总人数N=10，开始位置S=1，死亡数字M=1，剩余人数K=5

**实验结果截图：**

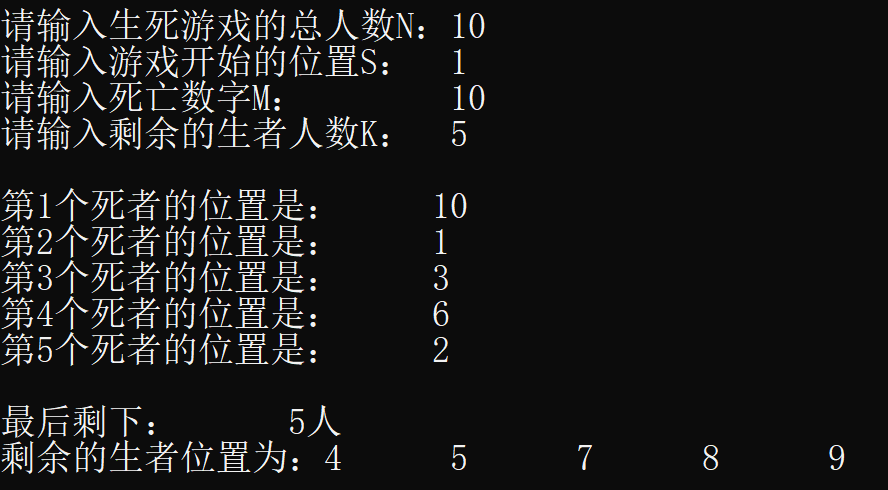


**实验结论**：程序正常运行且结果正确

4.2.2 死亡数字为N（第一次即删除尾结点）时

**测试用例：**总人数N=10，开始位置S=1，死亡数字M=10，剩余人数K=5

**实验结果截图：**



**实验结论**：程序正常运行且结果正确

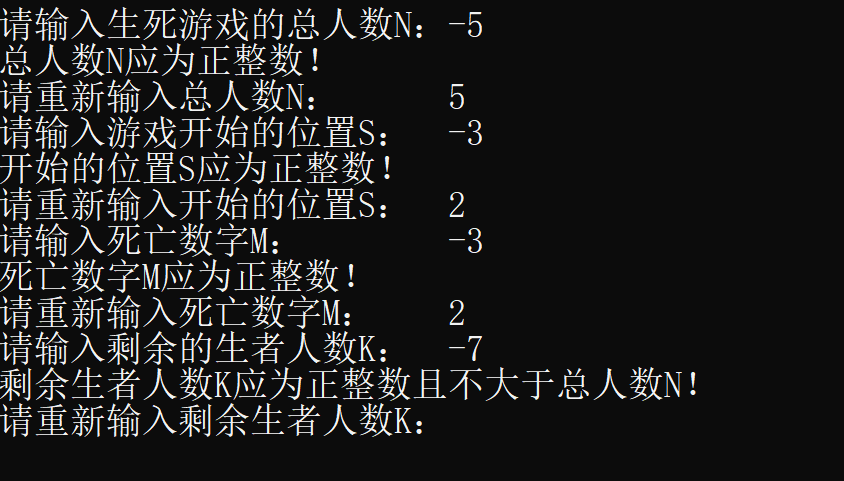
4.3 出错测试

4.3.1 输入为负数

**测试用例：**输入人数、位置、死亡数字、剩余人数等数据为负数

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**符合预期



4.3.2 输入为字母

**测试用例：**输入人数、位置、死亡数字、剩余人数等数据为非数字

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**符合预期

