项目说明文档

数据结构课程设计

——约瑟夫生死游戏

作 者 姓 名： 陈君

学 号： 2250420

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji Universit

# 1 分析

## 1.1 背景分析

约瑟夫生者死者游戏，也被称为“约瑟夫问题”或“约瑟夫环”，是一个著名的数学问题。该问题可以描述为：一群人围成一个圆圈，从某个位置开始数数，每数到第M个人，就将这个人杀掉，然后从下一个人重新开始数数，如此循环下去，直到圈内所有人都被杀掉。问题是，给定总人数N和步长M，求出哪个位置的人是最后的生者。

尽管约瑟夫问题起源于历史事件，但它在计算机科学、数学和其他领域都有广泛的应用。例如，该问题与数据结构中的环形链表相关，也与分布式系统中的问题相关。

总的来说，约瑟夫问题是一个深受数学家和计算机科学家喜欢的经典问题，不仅因为它有趣，而且它具有实际应用的价值。

## 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

如上功能分析所述，该游戏要求大量的增加、删除、修改操作，而链表进行增加、删除等操作十分简便，同时这是一个环形结构的数学模型，因此考虑使用单循环链表数据结构。

## 2.2 类结构设计

本系统采用设计listnode和List两个类，并且把List作为listnode类的友元，这样使得链表类（List）可以访问链表结点。

## 2.3 成员与操作设计

**链表结点类（listnode）**

**私有成员：**

Int num;//记录这个人的位置

listnode\* next;//记录下一个结点的地址

**公有操作：**

listnode(int n);//构造函数

listnode();//默认构造函数

int getnum()

{return num; }//获取结点位置

**链表类（LinkList）**

**私有成员：**

listnode\* first, \* last; //表头和表尾指针

**公有操作：**

List(int n);//构造函数

List();//默认构造函数

~List() ;//析构函数

List(int n);//构造函数

List();//默认构造函数

void count(int start, int dead\_num, int survior\_num);//根据所给数据开始游戏

void putout();//输出幸存者

## 2.4 系统设计

系统给出提示“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，再由下一个人开是报数，如此循环，直至剩下K个人为止”，然后提示用户输入游戏所需要的参数“请输入生死游戏的总人数N：请输入游戏开始的位置S： 请输入死亡数字M：请输入剩余的死者人数K：”，接着调用List的构造函数，接着调用count函数开始游戏，最后调用putout函数输出“剩余的生者位置“ 。

# 3 实现

## 3.1 List(int n);//构造函数功能的实现

具体实现，首结点，new命令创建新结点，输入位置值，将这个结点地址赋值给first，和last，将last结点next的值赋值为first，实现循环链表。后面的for循环循环n-1次，每次用new命令创建新结点，输入位置值，将新节点地址赋值给last结点的next，然后last=newnode，将last结点next的值赋值为first，实现循环链表。

### 3.2.3 构造函数功能截屏示例

## 3.2 count功能的实现

具体实现：首先使用for循环将指针跳转到start位置，然后for循环N-K次，代表杀了这么多人，for循环中嵌套一个for循环，循环M-2次，跳转到要删除结点的前一个结点，然后执行删除操作：新建一个q指针存放要删除的结点，将q指针的后一个结点连接到p指针所指的结点，同时判断特殊情况，当q是头节点或者是尾结点时，应该将first和last指针更新，最后输出q结点存储的位置然后再把q结点删除。

i

### 3.2.3 count功能截屏示例

## 3.3 putout功能的实现

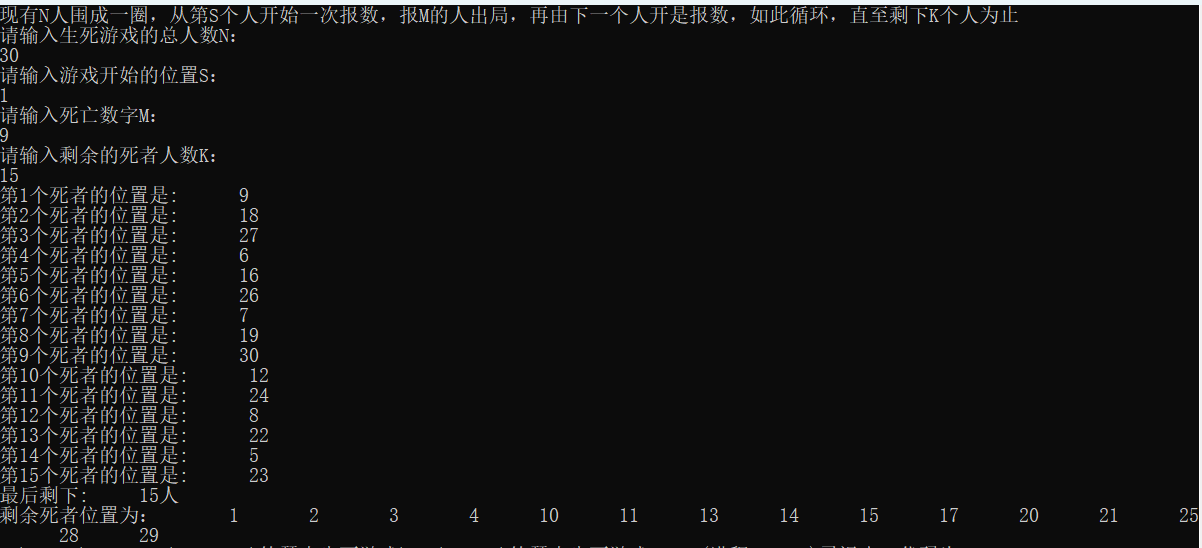
从头结点开始，打印结点中存储的位置，然后将p结点指向下一个位置，知道p结点的下一个结点为first为止。

### 3.3.2 putout功能截图示例

## 3.6 总体系统的实现

系统给出提示“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，再由下一个人开是报数，如此循环，直至剩下K个人为止”，然后提示用户输入游戏所需要的参数“请输入生死游戏的总人数N：请输入游戏开始的位置S： 请输入死亡数字M：请输入剩余的死者人数K：”，接着调用List的构造函数，接着调用count函数开始游戏，最后调用putout函数输出“剩余的生者位置“ 。

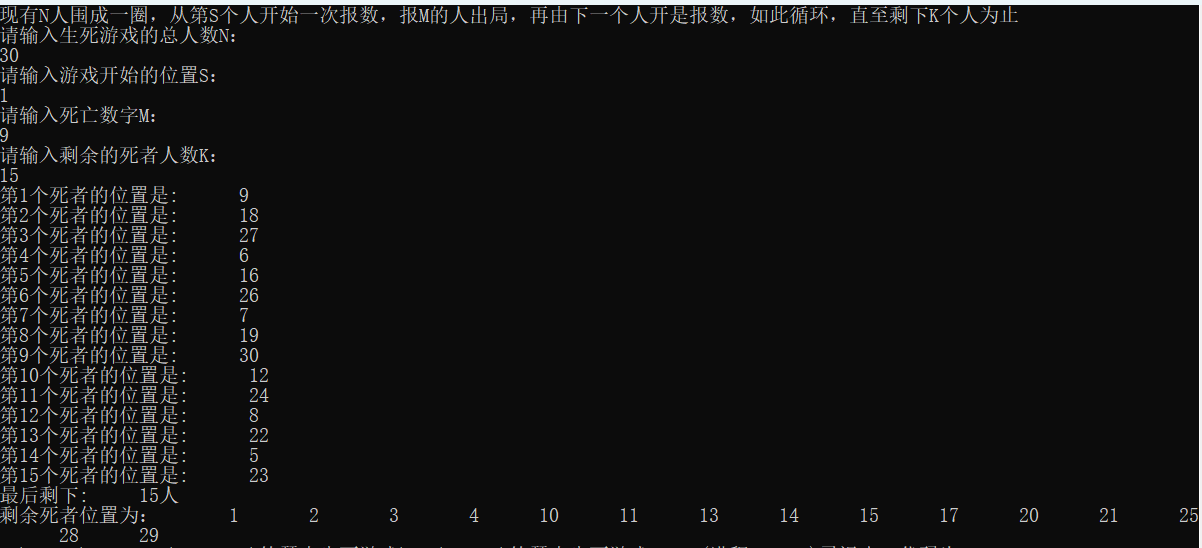
### }3.6.1 总体系统截屏示例

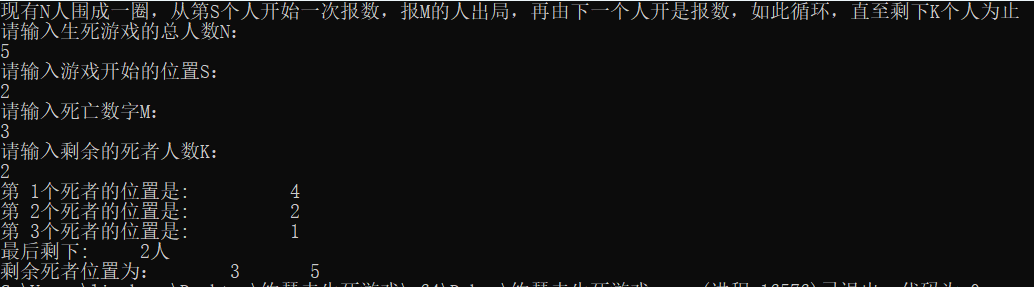


# 4 测试

## 4.1 游戏总体功能测试功能测试

### 4.1.1 截屏展示



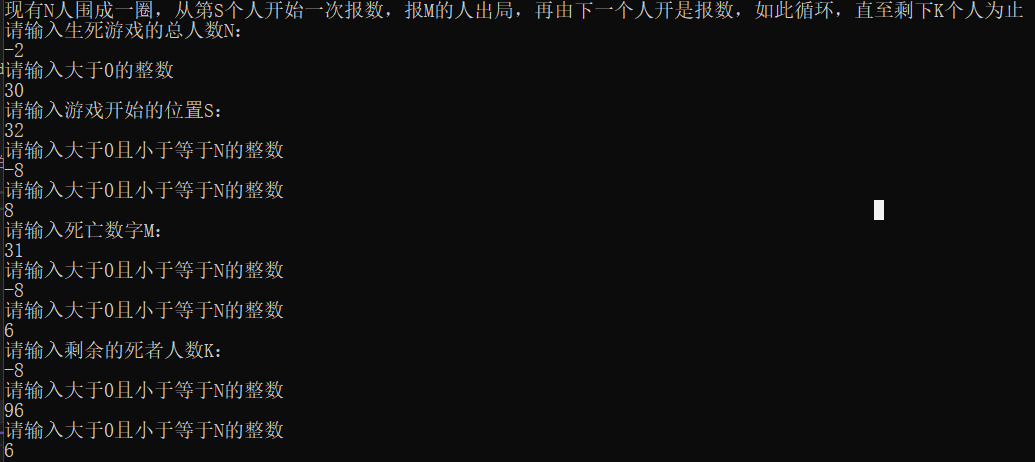


## 4.2 错误测试

### 4.2.1 初始化输入错误数据

**测试用例：**初始化中错误输入数据

**预期结果：**给出错误提示，程序运行正常不崩溃。

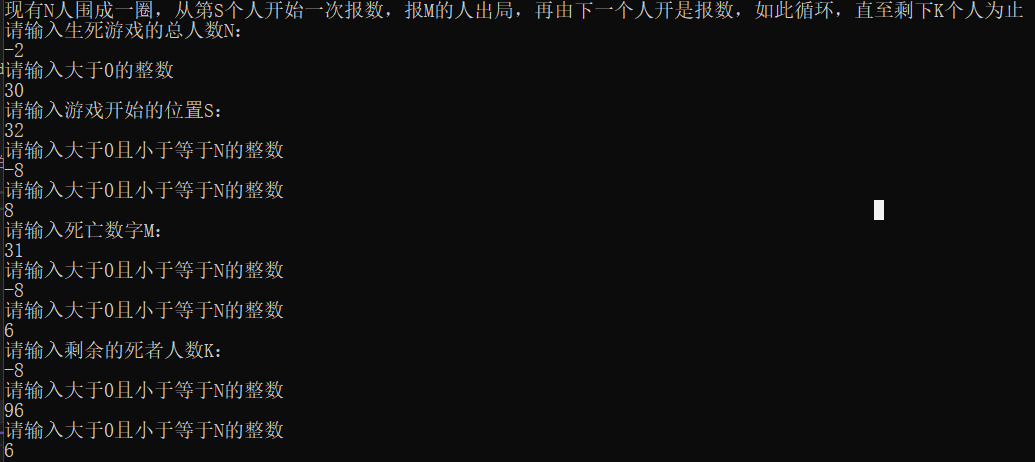
**实验结果：**

## 4.3 出错测试

### 4.3.1 总人数错误，死亡人数输入错误，开始位置错误，剩余的生者人数错误

**测试用例：**输入错误信息

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

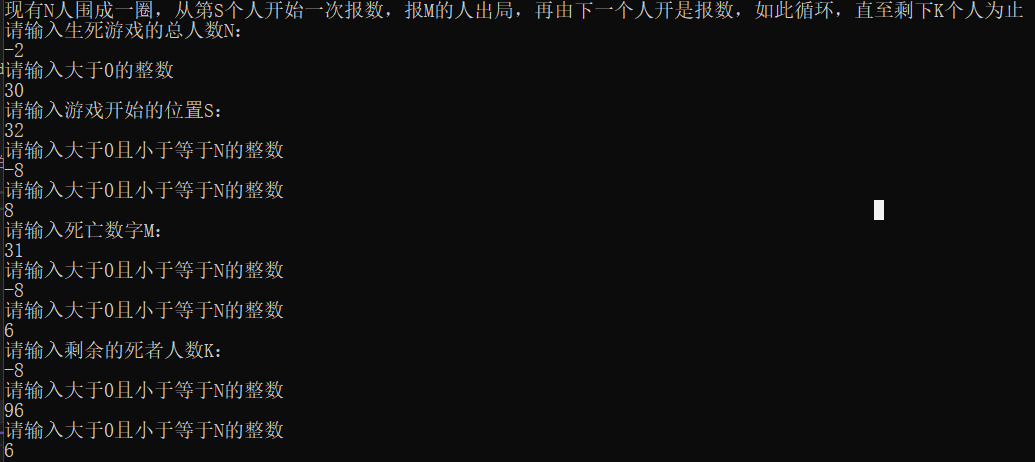
**实验结果：**

### 4.3.2 死亡人数输入错误

**测试用例：**死亡人数比总人数多

**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：**

****