数据结构课程设计

项目说明文档

**约瑟夫生者死者游戏**



同济大学

Tongji University

姓名： 林觉凯

学号： 2253744

指导老师： 张颖

学院专业： 软件学院 软件工程

**目录**

**1.项目分析-------------------------------------------------------------------------3**

**1.1 项目背景分析-------------------------------------------------------------------------3**

**1.2项目功能分析--------------------------------------------------------------------------3**

**1.2.1项目功能要求-------------------------------------------------------------------------------3**

**1.2.2项目输入要求-------------------------------------------------------------------------------3**

**1.2.3项目输出要求-------------------------------------------------------------------------------3**

**1.2.4项目实例-------------------------------------------------------------------------------------3**

**2.项目设计-------------------------------------------------------------------------4**

**2.1 数据结构设计-------------------------------------------------------------------------4**

**2.2 结构体设计----------------------------------------------------------------------------4**

**2.3系统流程设计-------------------------------------------------------------------------4**

**3.项目实现------------------------------------------------------------------------5**

**3.1 get\_number函数的实现-----------------------------------------------------------5**

**3.2 Algorithm函数的实现-------------------------------------------------------------5**

**3.3 main函数的实现--------------------------------------------------------------------7**

**4.项目代码功能测试------------------------------------------------------------8**

**4.1数据输入错误测试-------------------------------------------------------------------8**

**4.2 游戏开始位置、死亡数字和幸存人数输入错误测试-----------------------8**

**4.3游戏数据测试------------------------------------------------------------------------9**

**5.项目心得与体会--------------------------------------------------------------10**

**1.项目分析**

**1.1 项目背景分析**

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项目功能分析**

**1.2.1项目功能要求**

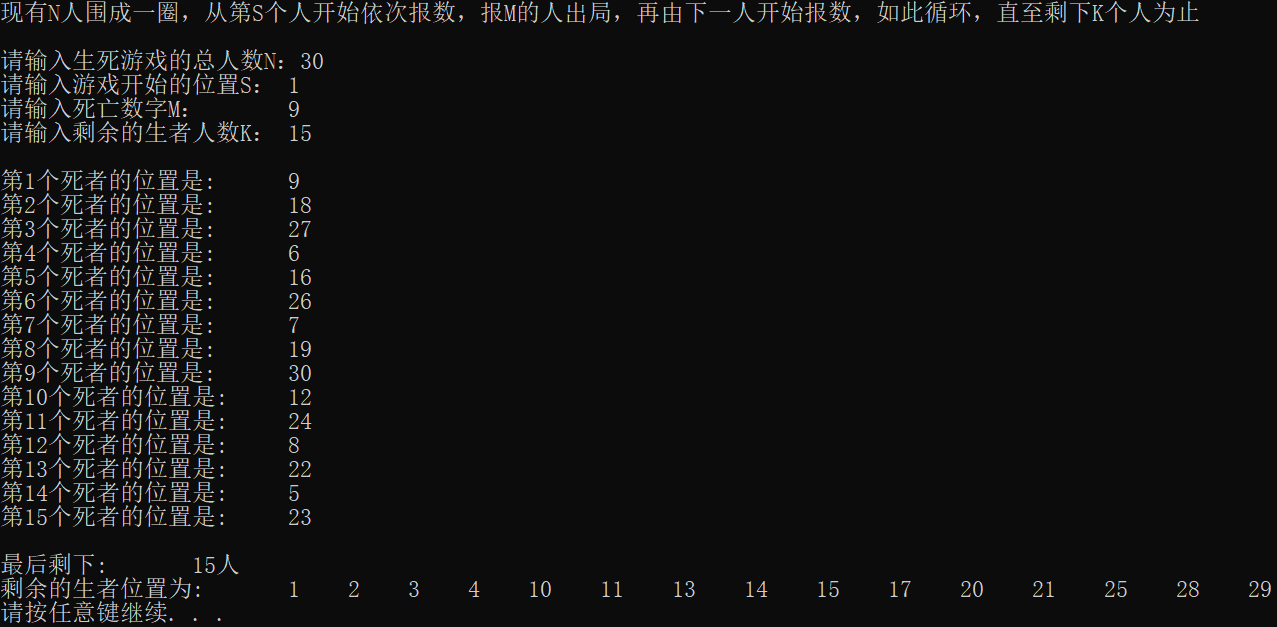
本项目的功能要求为使用循环单链表完成对约瑟夫生死环的模拟，并且在给定的总人数、开始位置、报数数值和剩余人数的情况下可以解出离开旅客的位置和剩余生者的位置。

**1.2.2项目输入要求**

输入约瑟夫生死游戏的总人数、开始位置、报数数值和剩余人数。

**1.2.3项目输出要求**

依次输出每一位离开旅客的位置和最后所有剩余生者的对应位置。

**1.2.4项目示例**

**2.项目设计**

**2.1 数据结构设计**

本题目要求使用循环单链表来实现约瑟夫问题的模拟。该系统频繁地进行结点的删除工作，所以运用链表可以方便地进行这些操作，我们只需通过相应函数找到要删除链表节点的地址，对其进行操作即可，不影响别的数据。同时，由题目易知我们还需要将尾结点与头结点相连，这样才能做到循环操作，不断遍历下去。所以本程序采用循环单链表这一数据结构进行实现。

**2.2结构体设计**

结构体是一种特殊的数据类型，它是具有一系列相同类型或者不同类型的数据集合。本道题较第一题来说需要用到的操作函数较少，数据类型较少，故本题采用链表和结构体来完成题目的解答。

结构体node可以代表这每一位旅客，它由两个不同的数据类型组成：这个结点(旅客)的位置(int)number和下一个结点(旅客)的地址(node\*)next。

**2.3系统流程设计**

系统操作的整体流程大致如下：

**程序开始运行**

**由get\_number函数得到**

**N S M K**

**Algorithm函数**

**建立模型,求解问题**

**显示最后结果**

**3.项目实现**

**3.1 get\_number函数的实现**

//获得一个在所需范围内的数字的函数

int get\_number(const int min = INT\_MIN, const int max = INT\_MAX, string errortip = "")

{

int number;

while (1)

{

cin >> number;

if (cin.good() == 0 || number<min || number>max)

//1.判断输入的是不是数字 2.判断输入的数字是否符合想要的范围

{

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << errortip; //如果输入的不是一个所需范围内的数字，则会输出提示errortip

continue;

}

break;

}

return number; //获得一个合理范围数字的函数，简化后续的代码健壮性检验

}

get\_number函数的作用是获得一个大小在所需范围内的数字，我们可以通过输入这个数字范围的最大值和最小值，同时写下如果输入错误时的提示语句，就可以利用函数里的大小判断和数据类型判断返回一个正确的合适范围内的数字。利用这个函数我们既完成了代码健壮性的任务。又可以通过在需要时调用这个函数实现代码的简洁性。

**3.2 Algorithm函数的实现**

void Algorithm(int N, int S, int M, int K)

{

Node\* head = NULL, \* current = NULL, \* temp = NULL, \* operation = NULL; //开始定义四个指针,head指针为头指针,current指针为当前指针,temp指针用于操作时的临时运用,operation指针用于删除链表中的元素

head = (Node\*)malloc(sizeof(Node)); //头结点申请一片空间

if (head == NULL)

{

cout << "链表创建失败！" << endl;

return;

} //代码的健壮性,如果内存的申请分配失败则会报告

head->number = 1; //从1开始编号,故头指针的编号为1

head->next = NULL; //目前只有一个链表结点,所以head的next域指向NULL

current = head;

//由于头指针不能变化,先让current指针等于头指针,找到链表开始位置,方便后续的循环操作

for (int i = 2; i <= N; i++)

{

temp = (Node\*)malloc(sizeof(Node));

if (temp == NULL)

{

cout << "链表创建失败！" << endl;

//代码的健壮性,如果内存的申请分配失败则会报告

system("pause");

return;

}

temp->number = i;

temp->next = NULL;

current->next = temp;

current = temp;

} //运用循环操作的尾插法来创建链表剩下的N-1个结点

current->next = head; //创建循环链表,让最后一个结点的next域指向头结点

current = head; //为了后续的操作方便,先将current指针指向头结点

for (int i = 1; i < S; i++)

{

current = current->next;

} //从指定的位置第S号开始游戏,该操作完成了将current指针指向第S号位置的操作

int remainer = N; //设置参数幸存者remainer,以作为下列while循环的判断条件

while (remainer > K)

//判断条件:幸存者remainer的值大于最后剩下的人数K,则该while循环进行

{

if(M > 1)

{

for (int i = 1; i < M; i++)

{

temp = current;

current = current->next;

}

}

else

{

temp = current; //对M等于1时进行特判，即每次删除下一个结点

}

if (current == head)

//如果要被删除的是头结点，那就把下一个结点设为头结点

{

head = current->next;

}

cout << "第" << N - remainer + 1 << "个死者的位置是:\t" << current->number << endl;

if (temp != NULL)

{

temp->next = current->next;

}

operation = current;

current = current->next;

delete operation;

remainer--; //删除指定的结点

}

cout << endl;

cout << "最后剩下:\t" << K << "人" << endl;

cout << "剩余的生者位置为:\t";

current = head;

for (int i = 0; i < K; i++) //遍历链表，输出最后生者的位置

{

cout << current->number << " ";

current = current->next;

}

}

Algorithm函数的主体分为三个部分。第一，构建链表模型。我们通过尾插法来将参加游戏的旅客基本信息的链表结点连接起来，构建单循环链表；第二，找到游戏开始位置。我们通过current指针的不断移动来找到开始位置S；第三，删除指定的链表结点，循环并完成操作。最后输出答案。注意，当M=1时要进行特判，因为我们的循环是从i=1开始的，所以当M=1时，我们需要让temp直接等于current，否则会导致temp是空指针导致错误。

**3.3 main函数的实现**

int main()

{

int N, S, M, K;

cout << "现有N人围成一圈，从第S个人开始依次报数，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" << endl; //给出约瑟夫生死环游戏提示

cout << endl;

cout << "请输入生死游戏的总人数N：";

N = get\_number(0, INT\_MAX, "输入错误!请输入一个正整数, 请重新输入："); //使用get\_number函数获得一个合理的N表示游戏的总人数

cout << "请输入游戏开始的位置S：\t";

S = get\_number(1, N, "输入错误!请输入一个大于1并小于N的正整数, 请重新输入："); //使用get\_number函数获得一个合理的S表示游戏的开始位置

cout << "请输入死亡数字M：\t";

M = get\_number(0, N, "输入错误!请输入一个大等于1并小于N的正整数, 请重新输入："); //使用get\_number函数获得一个合理的M表示游戏的报数数值

cout << "请输入剩余的生者人数K：\t";

K = get\_number(1, N - 1, "输入错误!请输入一个大等于1并小于N-1的正整数, 请重新输入："); //使用get\_number函数获得一个合理的K表示游戏的剩余人数

cout << endl;

Algorithm(N, S, M, K); //使用Algorithm函数获得最后的结果

system("pause");

return 0;

}

main函数主要是进行该游戏参数的输入工作。从main函数里调用get\_number函数来进行N，S，M，K数据的输入，再利用Algorithm函数进行问题的处理，最后得到结果。

**4.项目代码功能测试**

**4.1 数据输入错误测试**

**测试用例：**输入总人数为非正数或不为int型

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

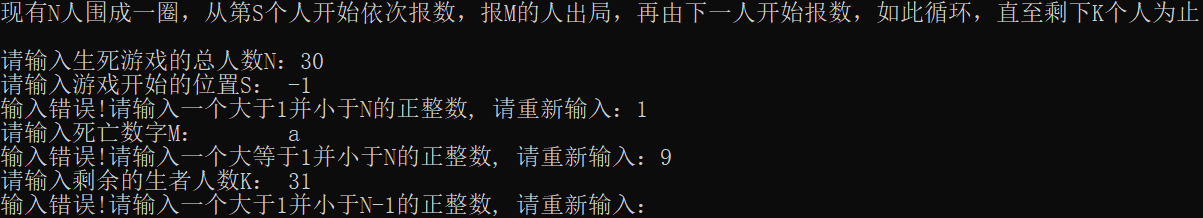
**实验结果：**

#### **c4.2**游戏开始位置，死亡数字和幸存人数输入错误测试

**测试用例：**输入总人数为非正数或不为int型或大于总人数N

**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：**

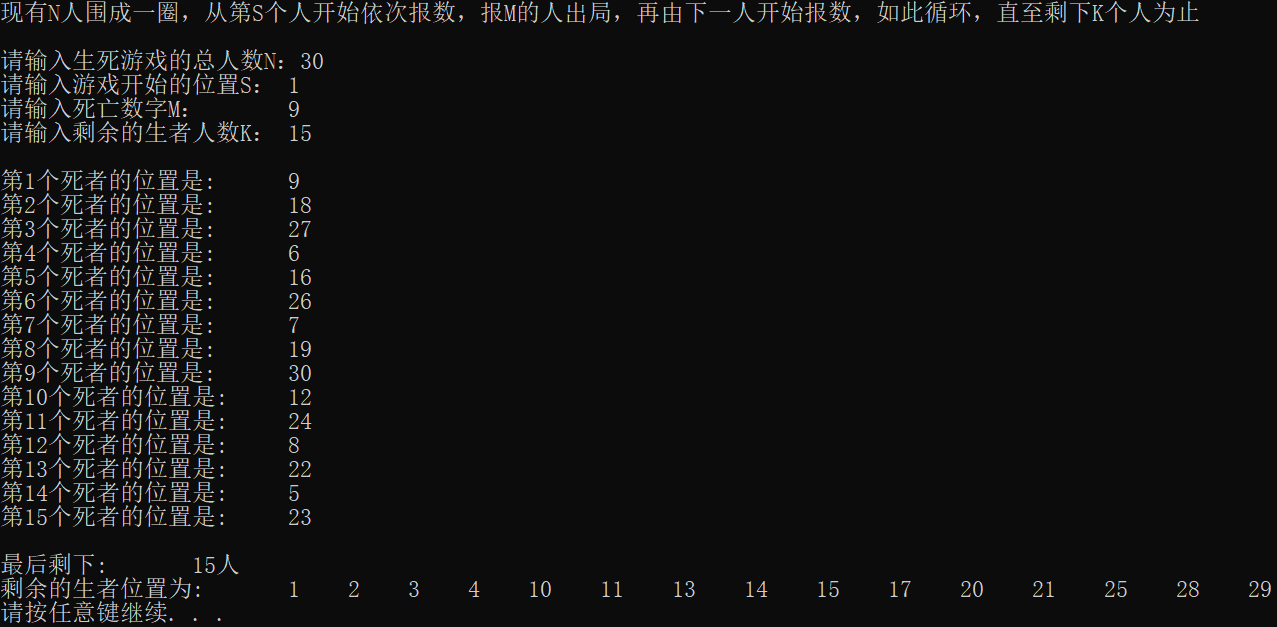
****

#### **4.3游戏数据测试**

**测试用例一：**30 1 9 15

**预期结果：**程序正常运行并给出正确答案

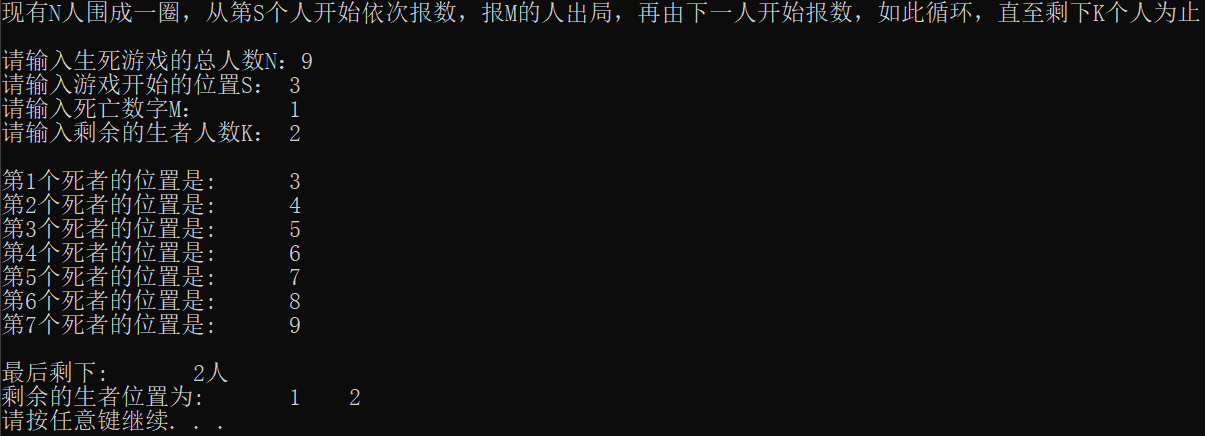
**实验结果：**

****

**测试用例二：**9 3 1 2 (M=1的特殊性判断样例)

**预期结果：**程序正常运行并给出正确答案

**实验结果：**

****

**5.项目心得与体会**

本次题目较上一个考试报名系统的创建相对来说比较简单，但是其中蕴含的链表的创建、删除结点和寻找链表位置的思想都大致可循。这道题目较上一题的区别就是本题构造的是一个单循环链表，所以我们在一些特殊性判断的位置要以头指针来确定循环条件的开始和结束。我在本次作业的书写过程中巩固了链表的构建以及遍历等基础操作，同时写了一个健壮性检验数字的函数，既保证了代码的简洁性又保证了代码的健壮性。总之，我在这次项目中巩固了我对链表知识点的理解和找到了优化代码健壮性的处理方式。