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： 杨宇琨

学 号： 2252843

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

# 1 分析

1. 数据结构选择：

- 旅客的序号可以使用单循环链表来存储，每个节点表示一个旅客，节点中包含旅客的序号以及指向下一个旅客的指针。

2. 算法设计：

- 创建一个包含N个旅客的单循环链表，每个节点表示一个旅客。

- 从指定的起始旅客开始，沿着链表进行循环遍历，每次数到第M个旅客就将其移出链表，直到剩余K个旅客为止。

- 在移出旅客时，需要更新链表的指针，确保链表的循环性质不受影响。

3. 用户交互：

- 用户输入旅客的个数N，离开旅客的间隔数M，以及指定的起始旅客序号。

- 程序根据用户输入创建单循环链表，并进行约瑟夫游戏的模拟计算。

- 输出离开旅客的序号和剩余旅客的序号。

4. 测试和优化：

- 进行测试，包括边界情况和异常输入的处理。

- 根据测试结果对程序进行优化，确保程序的正确性和稳定性。

# 2 设计

1. 数据结构设计

//Create a structure type to store the information and we want it to be a node in our linked list.

struct Player

{

int num;//The number of the player

bool alive;//Dead for false,alive for true

Player\* next;//Pointer points to the next player

};

- 结构体 `Player`：用于存储每个玩家的信息及链表节点的结构

- 成员变量：

- `int num`：玩家编号

- `bool alive`：玩家存活状态，true表示存活，false表示死亡

- `Player\* next`：指向下一个玩家的指针

1. 类结构设计

class PlayerList {

private:

Player\* head;

Player\* end; //A pointer points at the last player

public:

// Establish the list

PlayerList(int N);

// To delete all the space we created in this class

~PlayerList();

// Kill the people and display the position

void Display\_the\_death (int N, int S, int M, int K);

// respawn

void Reset();

};

- 类 `Player\_list`：

- 私有成员变量：

- `Player\* head`：指向链表头部的指针

- `Player\* end`：指向链表尾部的指针

- 公有成员函数：

- `Player\_list(int N)`：构造函数，用于创建链表和初始化玩家信息

- `void Display\_the\_death(int N, int S, int M, int K)`：游戏进行和结果展示的函数

- `void reset()`：重置游戏状态的函数

- `~Player\_list()`：析构函数，用于释放链表占用的内存

3.成员与操作设计

1. 构造函数

- 功能：用于建立链表

- 参数：N，旅客的个数

- 实现：在构造函数中创建一个包含N个玩家的循环链表，每个玩家的状态初始化为存活。

2. 析构函数

- 功能：释放链表占用的内存

- 实现：对链表中的每个节点进行删除操作，释放相应的内存空间。

3. Display\_the\_death方法

- 功能：进行约瑟夫游戏的计算，并输出结果

- 参数：N，旅客的个数；S，起始位置；M，离开旅客的间隔数；K，剩余旅客的个数

- 实现：根据输入的参数进行约瑟夫游戏的模拟计算，并输出离开旅客的序号和剩余旅客的序号。

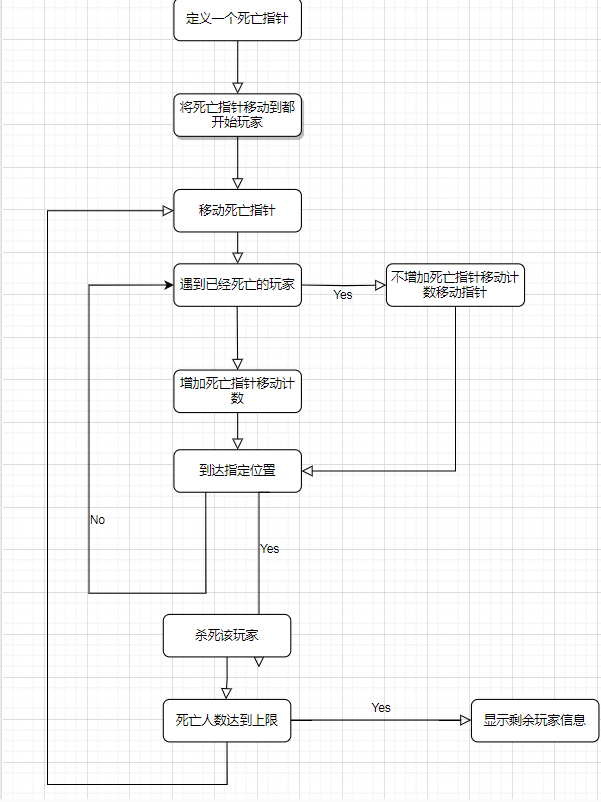
4. Reset方法

- 功能：重置链表，将所有玩家状态设为存活

- 实现：将链表中所有玩家的状态设为存活状态。

## 3.1 显示死者功能的实现

### 3.1.1 显示死者功能流程图



### 3.1.2 显示死者功能核心代码

void Display\_the\_death(int N, int S, int M, int K)

{

Player\* Deathmark = head;//Create a pointer to kill the people

//Move the starter to the start position

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

Deathmark = Deathmark->next;

}

//Use a loop to kill

for (int i = 0; i < N - K; i++) {

for (int count = 0; count < M; ) {

Deathmark = Deathmark->next;

if (Deathmark->alive)//Only count the people alive

count++;

}

Deathmark->alive = false;//Kill the people

cout << "第" << i + 1 << "个死者的位置是：" << Deathmark->num << endl;//Display the information

}//Kill N-K people

//Display the settlement

cout << endl;

cout << std::left << std::setw(20) << "最后剩下" << K << "人" << endl;

cout <<"剩余生者位置为:" << endl;

//Display all the people that are still alive

for (Player\* current = head->next; current != end;) {

if (current->alive)

cout << std::right << std::setw(8) << current->num;

current = current->next;

}

if(end->alive)

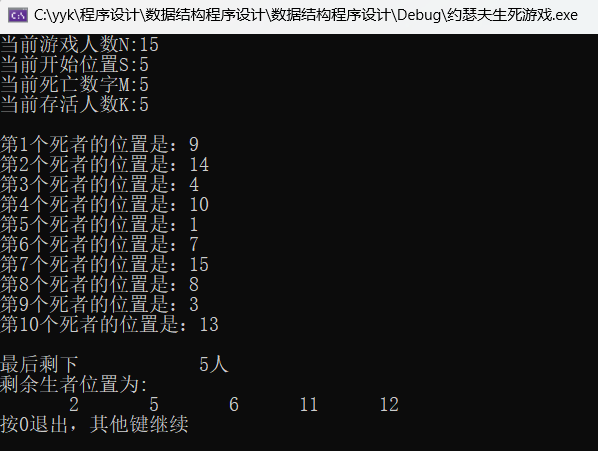
cout << std::right << std::setw(8) << end->num;

cout << endl;

}

### 

### 3.1.3 显示死者功能截屏示例



## 3.2 总体系统的实现

1. \*\*输入交互\*\*：

- 主函数通过命令行提示用户输入游戏的参数，包括游戏人数N、开始位置S、死亡数字M和存活人数K。

- 输入的合法性检查：程序对用户输入的参数进行合法性检查，确保输入在指定范围内。

2. \*\*游戏模拟\*\*：

- 主函数创建了一个名为`List`的`Player\_list`对象，用于存储玩家信息和进行游戏模拟。

- 在用户输入参数后，调用`List`的`Display\_the\_death`方法进行约瑟夫游戏的模拟计算，并输出游戏结果。

- 游戏结果显示：程序会清空屏幕，显示当前的游戏参数，并输出经过约瑟夫游戏后的存活玩家位置，然后重置游戏状态。

3. \*\*用户交互\*\*：

- 用户可以选择继续游戏或者退出程序。

- 选择继续游戏时，程序将清空屏幕，重新提示用户输入游戏参数，然后进行游戏模拟。

- 选择退出程序时，程序将显示"程序结束！"并退出。

4. \*\*异常处理\*\*：

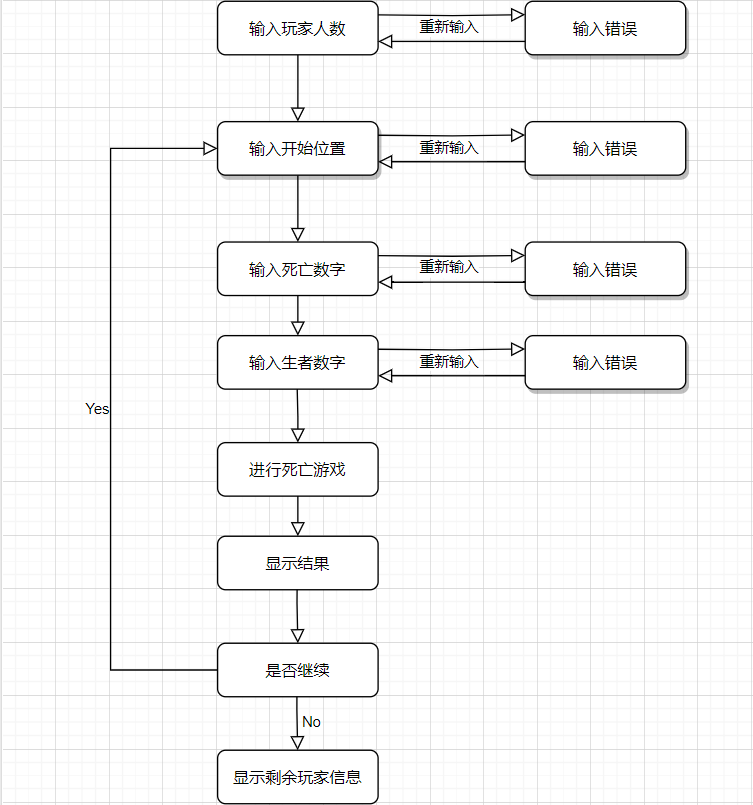
- 在用户输入、游戏进行过程中，程序需要对异常情况进行处理，包括输入错误、游戏参数错误等情况。

5. \*\*结构设计\*\*：

- 主函数采用了循环结构，使得用户可以反复进行游戏，直到选择退出程序。

- 主函数中的用户输入、游戏模拟、显示结果等功能都通过调用`Player\_list`类中的方法来实现。

### 3.2.1 总体系统流程图



### 3.2.2 总体系统核心代码

int main()

{

int N, S, M, K;

//Display the prompt and let the client input

cout << "请输入游戏人数(10-50)" << endl;

N = input<int>(N\_l\_limit, N\_U\_limit);

//Create a new list

Player\_list List(N);

while (true) {

//Clear our screen

system("cls");

std::cout << "请输入开始位置(1-" << N << ")" << endl;

S = input<int>(S\_l\_limit, N);

cout << "请输入死亡数字(0-" << N << ")" << endl;

M = input<int>(M\_l\_limit, N);

std::cout << "请输入剩余生者数(0-" << N << ")" << endl;

K = input<int>(K\_l\_limit, N);

//Clear our screen

system("cls");

//Display our condition

std::cout << "当前游戏人数N:" << N << std::endl;

std::cout << "当前开始位置S:" << S << std::endl;

std::cout << "当前死亡数字M:" << M << std::endl;

std::cout << "当前存活人数K:" << K << std::endl;

cout << endl;

List.Display\_the\_death(N, S, M, K);

List.reset();

//Display the prompt

std::cout << "按0退出，其他键继续" << std::endl;

char controller;//Create a variable to hold the choice we input

controller = \_getch();//\_getch can avoid displaying undesired input on the screen

if (controller == '0') {

std::cout << "程序结束！" << std::endl;

return 0;

}

}//end of while

return 0;

### } 3.2.3 总体系统截屏示例

