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：陆诚彬

学　　号：2254321

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 项目背景

本项目设计报告的核心是“约瑟夫生者死者游戏”，一种源于历史传说的游戏。据传，30 名旅客由于船只超载，在风高浪大的情况下，被迫通过一种残酷的方式决定谁能幸存。船长宣布，只有将半数旅客投入海中，剩余人员才可能生还。在这种绝望的情境中，旅客们决定围成一圈，依次报数，每当数到第 9 个人时，就将其扔入海中，直至剩下半数旅客。

这个过程形成了一个有趣的数学问题：在围成一圈的 N 个人中，从某个点开始数数，每数到第 M 个人就将其移除，如此循环，直到剩下 K 个人。该问题可以通过数学建模与程序设计解决，成为了本项目的数学核心。

2 项目需求分析

2.1 功能需求

1) **数据结构设计**：采用单循环链表来模拟旅客围成一圈的情形，有效地处理元素的动态添加和删除。
2) **数学建模与算法实现**：基于用户输入的 N 和 M，程序应能准确模拟并实现约瑟夫生者死者游戏的逻辑，即从指定的起始点开始，每数到第 M 个人时将其移除，直到剩下一定数量的人。

2.2 非功能需求

1) 程序设计需要考虑到算法效率与数据结构的合理应用。
2) 用户界面应简洁明了，易于用户输入数据并获取结果。
3) 代码应具有良好的可读性和注释，便于理解和维护。
4) 可视化演示：考虑增加一个简单的图形界面，直观地展示游戏过程。
5) 变体探索：探讨不同的 M 值和起始点对最终结果的影响，增加游戏的趣味性和教育意义。

2.3 项目输入输出需求

2.3.1 输入格式

生死游戏的总人数(N)、游戏开始的位置(S)

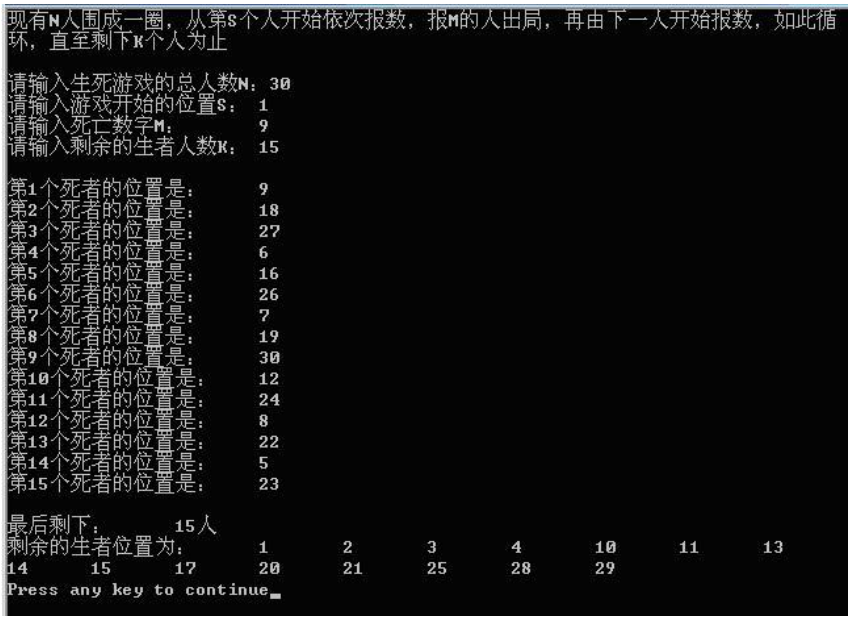
死亡数字(M)、剩余的生者人数(K)

2.3.2 输出格式

离开旅客序号：按顺序输出被移除的旅客编号。

剩余旅客序号：在游戏结束时，输出最终幸存的旅客编号。

2.3.3 项目示例



3 项目设计

3.1 数据结构设计

由于本项目比较简单，故并没有采用 OOP 的方式封装类，而是采用结构体和函数的组合来实现约瑟夫游戏的逻辑。

3.1.1 单循环链表

在约瑟夫生者死者游戏中，我们采用单循环链表来模拟旅客围成一圈的情况。单循环链表是一种特殊的链表结构，其中链表的尾节点指向头节点，形成一个环状结构。这种结构特别适用于约瑟夫问题，因为它可以无缝地处理环形队列的遍历和元素的删除。

3.1.2 节点结构体设计

Node 结构体：每个节点包含一个 int 类型的 number 成员变量，用于存储旅客的编号，以及一个指向下一个节点的 Node* next 指针。

```
struct Node {
    int number;
    Node* next;
    Node(int num) : number(num), next(nullptr) {}
};
```

3.1.3 链表创建和遍历

创建单循环链表：createCircularList 函数接受一个整数 N 作为输入，表示链表的节点数量。函数初始化一个头节点，并循环创建 N-1 个其他节点，将它们链接起来，并将最后一个节点的 next 指针指向头节点，形成一个闭环。

链表遍历：为了实现约瑟夫游戏逻辑，我们需要遍历这个单循环链表，每次移动 M 步来找到需要删除的节点，并对链表进行相应的节点删除操作。

4 项目实现

4.1 主函数实现

4.1.1 主函数功能简介

功能：程序的入口点，处理用户输入并调用游戏逻辑。

流程：

1) 提示用户输入总人数(N)，开始位置(S)，死亡数字(M)以及剩余生者人数(K)。
2) 调用 josephusGame 函数执行游戏逻辑。
3) 程序暂停，等待用户操作。

4.1.2 主函数核心代码

```
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，再又下一人开始报数，如此循环，直至剩下 K 个人为止" << endl;
    cout << "请输入生死游戏的总人数(N): ";
    cin >> N;
    cout << "请输入游戏开始的位置(S): ";
    cin >> S;
```

```
        cout << "请输入死亡数字(M): ";
        cin >> M;
        cout << "请输入剩余的生者人数(K): ";
        cin >> K;
        josephusGame(N, S, M, K);

        system("pause");

        return 0;
    }
```

4.2 单循环链表创建实现

4.2.1 单循环链表创建功能简介

功能：创建包含 N 个节点的单循环链表。

参数：旅客的总数 N。

返回值：指向链表头节点的指针。

流程：

1) 创建头节点并初始化。
2) 循环创建其他节点，并将它们连接起来。
3) 最后一个节点连接到头节点，形成环形。

4.2.2 单循环链表创建核心代码

```
    Node* createCircularList(int N) {
        Node* head = new Node(1);
        Node* current = head;
        for (int i = 2; i <= N; ++i) {
            current->next = new Node(i);
            current = current->next;
        }
        current->next = head; // 形成环形
        return head;
    }
```

4.3 打印剩余旅客编号实现

4.3.1 打印剩余旅客编号功能简介

功能：打印剩余旅客的编号。

参数：指向链表头节点的指针 head，剩余旅客数 K。

流程：

1) 遍历链表，收集剩余旅客的编号。
2) 对收集到的编号进行排序（可选）。
3) 打印排序后的编号。

4.3.2 打印剩余旅客编号核心代码

```
1.    void printRemaining(Node* head, int K) {
2.        int* remaining = new int[K];
3.        Node* current = head;
4.        for (int i = 0; i < K; ++i) {
5.            remaining[i] = current->number;
6.            current = current->next;
7.        }
8.
9.        // 简单的冒泡排序
10.       for (int i = 0; i < K - 1; i++) {
11.           for (int j = 0; j < K - i - 1; j++) {
12.               if (remaining[j] > remaining[j + 1]) {
13.                   int temp = remaining[j];
14.                   remaining[j] = remaining[j + 1];
15.                   remaining[j + 1] = temp;
16.               }
17.           }
18.       }
19.
20.       cout << "剩余的生者位置为: ";
21.       for (int i = 0; i < K; ++i) {
22.           cout << remaining[i] << " ";
23.       }
24.       cout << endl;
25.
26.       delete[] remaining;
27.   }
```

4.4 约瑟夫生者死者游戏逻辑实现

4.4.1 约瑟夫生者死者游戏逻辑功能简介

功能：实现约瑟夫游戏的主要逻辑。

参数：总人数 N，开始位置 S，死亡数字 M，剩余生者人数 K。

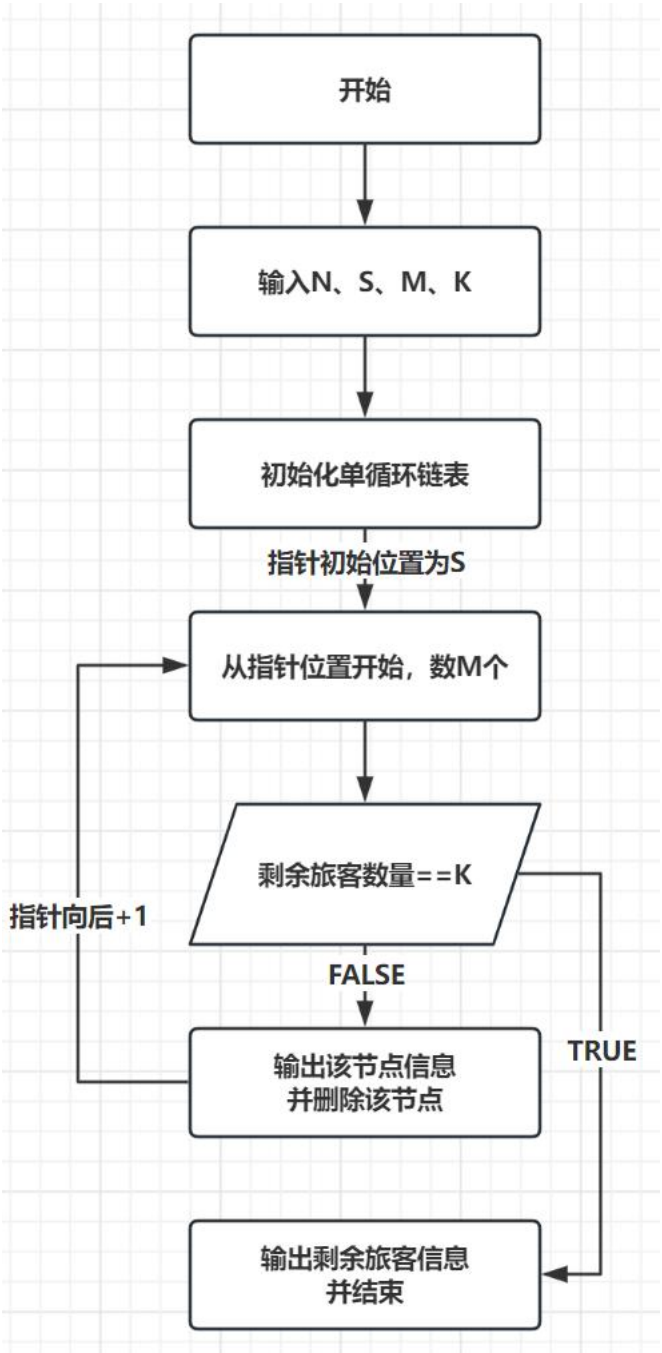
流程：

1) 创建单循环链表。
2) 从指定位置开始，依次数到 M 并删除相应节点。
3) 直至链表中剩余 K 个节点。
4) 调用 printRemaining 函数打印剩余旅客的编号。

4.4.2 约瑟夫生者死者游戏逻辑核心代码

```
1.    void josephusGame(int N, int S, int M, int K) {
2.        Node* head = createCircularList(N);
3.        Node* prev = head;
4.        for (int i = 1; i < S; ++i) {
5.            prev = head;
6.            head = head->next;
7.        }
8.        int cnt = 1;
9.        cout << endl;
10.       while (N > K) {
11.           for (int count = 1; count < M; ++count) {
12.               prev = head;
13.               head = head->next;
14.           }
15.           prev->next = head->next;
16.             cout << "第" << setw(2) <<cnt++ << "个人出局，出局编号
    为:          ";
17.           cout << setiosflags(ios::left);
18.           cout << setw(2) << head->number << endl;
19.           cout << resetiosflags(ios::left);
20.           delete head;
21.           head = prev->next;
22.           --N;
23.       }
24.       cout << endl;
25.       printRemaining(head, K);
26.   }
```

4.5 系统总体功能流程图



5 设计小结

本项目的设计和实现围绕“约瑟夫生者死者游戏”展开，成功地将一个古老的数学问题转化为了一个现代的程序设计任务。通过对项目需求的深入分析、合理的数据结构设计，以及明确的功能实现，本项目不仅在技术层面达到了预期的目标，而且在教育和趣味性方面也取得了良好效果。

5.1 主要优点

1) **数据结构的合理应用：**通过采用单循环链表，本项目有效地模拟了旅客围成一圈的场景，充分展现了链表在处理动态数据中的优势。
2) **清晰的功能实现：**项目通过分模块的方式实现了主要功能，包括链表的创建、遍历、节点的删除和剩余旅客编号的打印。每个模块都有明确的职责和清晰的接口，确保了代码的可读性和可维护性。
3) **用户交互的简洁性：**通过简单直观的控制台输入输出，用户可以轻松参与游戏，同时获得清晰的反馈，提升了用户体验。
4) **算法效率的考虑：**在实现游戏逻辑时，考虑了算法的效率，保证了程序运行的流畅性和稳定性。

5.2 改进空间

1) **图形界面的引入：**为了提升用户体验，可以考虑引入图形界面，使游戏更加直观和互动性更强。
2) **代码重构与面向对象设计：**虽然当前使用结构体和函数的组合实现了项目目标，但进一步采用面向对象的设计方式可以提高代码的模块化程度和可重用性。
3) **更丰富的游戏变体和参数设置：**增加更多的游戏设置选项，如不同的开始位置、报数间隔等，可以增加游戏的复杂性和教育意义。
4) **性能优化和异常处理：**进一步优化程序的性能，并增加异常处理机制，以应对非法输入和潜在的运行时错误。

总体而言，本项目在设计和实现上都达到了良好的标准，不仅为解决约瑟夫问题提供了一个有效的方法，也为程序设计和数据结构的学习提供了一个实际的案例。通过不断的迭代和改进，该项目有潜力成为一个更加完善和多功能的教育工具。

6.1 输入测试

6.1.1 正常输入

```
D:\DataStructure\02\02.exe
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，再又下一人开始报数，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)：20
请输入游戏开始的位置(S)：5
请输入死亡数字(M)：6
请输入剩余的生者人数(K)：9
```

6.1.2 输入超界/非法

```
请输入生死游戏的总人数(N)：aaa
错误：请输入一个正整数。
请输入生死游戏的总人数(N)：-99
错误：请输入一个正整数。
请输入生死游戏的总人数(N)：20
请输入游戏开始的位置(S)：50
错误：请输入一个正整数，且不能超过总人数。
请输入游戏开始的位置(S)：5
请输入死亡数字(M)：6
请输入剩余的生者人数(K)：25
错误：请输入一个非负整数，且不能超过总人数。
请输入剩余的生者人数(K)：5
```

结论：符合输入逻辑判断

6.2 输出测试

```
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，再又下一人开始报数，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)：20
请输入游戏开始的位置(S)：5
请输入死亡数字(M)：9
请输入剩余的生者人数(K)：6

第 1个人出局，出局编号为：          13
第 2个人出局，出局编号为：          2
第 3个人出局，出局编号为：          11
第 4个人出局，出局编号为：          1
第 5个人出局，出局编号为：          12
第 6个人出局，出局编号为：          4
第 7个人出局，出局编号为：          16
第 8个人出局，出局编号为：          8
第 9个人出局，出局编号为：          3
第10个人出局，出局编号为：          18
第11个人出局，出局编号为：          15
第12个人出局，出局编号为：          14
第13个人出局，出局编号为：          17
第14个人出局，出局编号为：          20

剩余的生者位置为：5 6 7 9 10 19
请按任意键继续. . .
```

结论：符合输出逻辑，且正确。