

# 目录

## 目录

### 1.项目背景

### 2.需求分析

### 3.设计思路

数据结构

功能实现

### 4.核心代码说明

节点类

链表类

约瑟夫环函数

函数接口说明

### 5.使用方法及函数功能那个演示

输入数据

输出数据

## 1.项目背景

本项目基于约瑟夫生死者游戏，约瑟夫生死者游戏的大意是：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。

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个人就让其出列，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. 离开旅客的间隔数，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

随后程序将输出：

1. 离开旅客的序号
2. 剩余旅客的序号

## 2.需求分析

本项目为实现约瑟夫生死者的游戏模拟，接受用户输入的游戏参数后，通过程序得出离开旅客的序号和剩余旅客的序号

## 3.设计思路

---

## 数据结构

约瑟夫生死者游戏的特点就是所有人的序列是一个循环序列，每次计数若超过最后一个人之后即跳转至第一个人，这样的特点选择单向循环链表作为数据结构是非常合适的，将每一位旅客作为一个链表节点

## 功能实现

采用带头结点的单向循环链表有序存储下所有旅客，用户数据输入完毕后，将链表指针的位置调整至游戏的起始位置，随后开始循环计数，每一次计数到达后，输出该节点旅客信息，随后执行删除节点操作，当删除的旅客数达到要求后，从头结点开始遍历链表，输出所有剩余旅客信息

## 4.核心代码说明

---

### 节点类

```
1 struct Node {
2     int position;
3     Node *pNext;
4     Node(int n):position(n),pNext(NULL){}
5     Node():position(0),pNext(NULL){}
6 };
```

用于存储单个旅客信息，包括位置（position），下个节点指针（\*pNext）

包含两个构造函数：默认构造函数，带位置的构造函数

### 链表类

```
1 //Create a circular link list to stimulate Joseph circle
2 class CircularLinkList {
3 private:
4     Node *head;
5     Node *tail;
6     int length;
7 public:
8     //Default constructor
9     CircularLinkList() {
10         head = new Node(0);
11     }
12     //Overload constructor
13     CircularLinkList(int amount);
14     //Delete node
15     void DeleteNode(Node *node);
16     //Remove people in terms of start and gap
17     void JosephStart(int start, int gap, int survival);
18 };
```

包含三个私有成员变量：头结点指针（\*head），尾节点指针（\*tail），长度（length）

成员函数的功能见注释

## 约瑟夫环函数

```
1 //Remove people in terms of start and gap
2 void JosephStart(int start, int gap, int survival) {
3     Node* temp = head;
4     //Adjust the pointer to start position
5     for (int i = 1; i <= start; i++) {
6         temp = temp->pNext;
7     }
8     while (length > survival) { //Keep recycling until the length less than that user entered
9         int k = 1;
10        //Adjust to the position one before required
11        for (int i = 0; i < gap-1; i++) {
12            temp = temp->pNext;
13        }
14        //Save the pointer
15        Node* delNode = temp;
16        temp = temp->pNext;
17        //Output information of the node
18        cout << "第" << k << "个死者的位置时: " << "\t" << delNode->postion << endl;
19        //Delete the node
20        DeleteNode(delNode);
21        k++;
22    }
23    //Output the survivals' information
24    cout << endl << "最后剩下" << survival << "人" << endl;
25    cout << "剩余生者的位置为: ";
26    Node* show = head->pNext;
27    for (int i = 0; i < survival; i++) {
28        cout << "\t" << show->postion;
29        show = show->pNext;
30    }
31    return;
32 }
```

首先调整指针位置到起始节点，然后在while循环中进行计数，一次到达要删除的节点，while循环终止的条件为剩余人数达到要求，在每一次遍历至待删除节点时，先输出其位置信息，随后调用 `DeleteNode` 函数删除该节点

```
1 //Delete node
2 void DeleteNode(Node *node) {
3     Node* temp = head;
4     while (temp->pNext->postion != node->postion && temp!= tail) {
5         temp = temp->pNext;
6     }
7     if (temp != tail) {
8         Node *delNode = temp->pNext;
9         temp->pNext = temp->pNext->pNext;
10        delete delNode;
11        length--;
12    }
13    else{
14        cout<<"Can't find the node!"<<endl;
```

```
15     }  
16     return;  
17 }
```

因为传入的 `node` 是待删除节点的前一个节点的指针，所以在该函数中需要重新遍历来找到待删除节点位置，然后删除

### 函数接口说明

| 返回值类型 | 成员函数名            | 参数                               | 属性     | 功能     |
|-------|------------------|----------------------------------|--------|--------|
| \     | CircularLinkList | \                                | public | 默认构造函数 |
| \     | CircularLinkLlst | (int amount)                     | public | 构造函数   |
| void  | DeleteNode       | (Node *node)                     | public | 删除节点   |
| void  | JosephStart      | (int start,int gap,int survival) | public | 约瑟夫环   |

## 5.使用方法及函数动能那个演示

双击 `joseph_circle.exe` 运行程序

### 输入数据

根据程序提示依次输入：

- 游戏总人数N
- 游戏开始的位置S
- 死亡数字M
- 剩余的生者人数K

G:\coding\Data Structure\数据结构课程作业\Data-Structure-Practice\exercise\_2\joseph\_circle.exe

现有N个人围成一圈，从第S个人开始依次报数，报M的人出局，再由下一个人开始报数，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： 30  
请输入游戏开始的位置： 1  
请输入死亡数字M： 9  
请输入剩余的生者人数K： 15

### 输出数据

输入完成后，程序将一次输出每一个死者位置，直至剩余人数达到用户输入人数，最后输出所有生者位置，游戏结束

```
G:\coding\Data Structure\数据结构课程作业\Data-Structure-Practice\exercise_2\joseph_circle.exe
现有N个人围成一圈，从第S个人开始依次报数，报M的人出局，再由下一个人开始报数，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: 30
请输入游戏开始的位置: 1
请输入死亡数字M: 9
请输入剩余的生者人数K: 15
第1个死者的位置时: 9
第1个死者的位置时: 18
第1个死者的位置时: 27
第1个死者的位置时: 6
第1个死者的位置时: 16
第1个死者的位置时: 26
第1个死者的位置时: 7
第1个死者的位置时: 19
第1个死者的位置时: 30
第1个死者的位置时: 12
第1个死者的位置时: 24
第1个死者的位置时: 8
第1个死者的位置时: 22
第1个死者的位置时: 5
第1个死者的位置时: 23
最后剩下15人
剩余生者的位置为: 1 2 3 4 10 11 13 14 15 17 20 21
25 28 29请按任意键继续. . .
```

按任意键退出程序，需要再次模拟则重复上述步骤