实验报告（Josephus 问题）

作者：林楠
时间：2023 年 9 月 25 日 *上海*

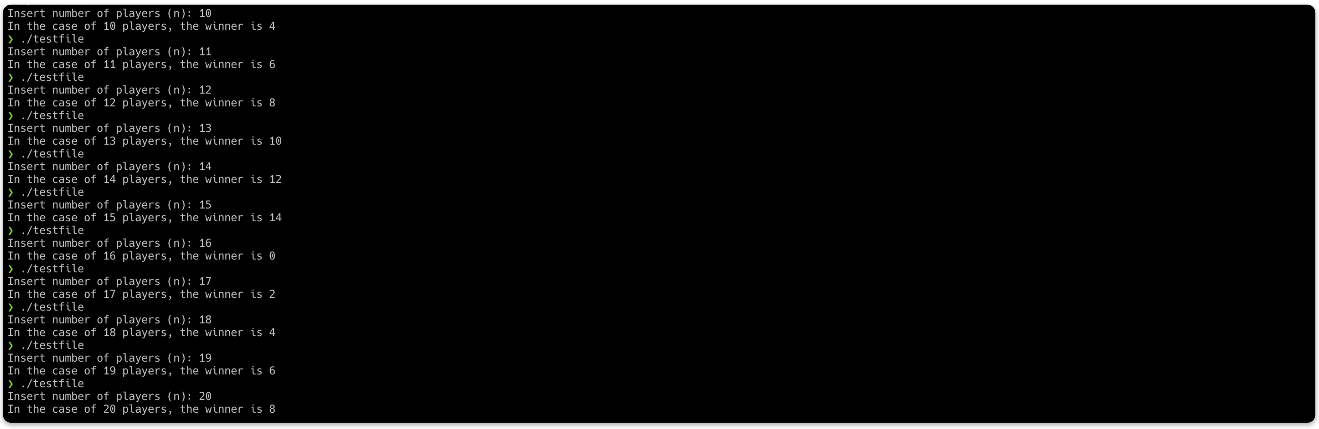
源码与结果

源码

见附录A

结果

n	Result
10	4
11	6
12	8
13	10
14	12
15	14
16	0
17	2
18	4
19	6
20	8



```
Insert number of players (n): 10
In the case of 10 players, the winner is 4
❯ ./testfile
Insert number of players (n): 11
In the case of 11 players, the winner is 6
❯ ./testfile
Insert number of players (n): 12
In the case of 12 players, the winner is 8
❯ ./testfile
Insert number of players (n): 13
In the case of 13 players, the winner is 10
❯ ./testfile
Insert number of players (n): 14
In the case of 14 players, the winner is 12
❯ ./testfile
Insert number of players (n): 15
In the case of 15 players, the winner is 14
❯ ./testfile
Insert number of players (n): 16
In the case of 16 players, the winner is 0
❯ ./testfile
Insert number of players (n): 17
In the case of 17 players, the winner is 2
❯ ./testfile
Insert number of players (n): 18
In the case of 18 players, the winner is 4
❯ ./testfile
Insert number of players (n): 19
In the case of 19 players, the winner is 6
❯ ./testfile
Insert number of players (n): 20
In the case of 20 players, the winner is 8
```

算法理解

定义

- 所有的玩家编号成一有序序列：$L_n = [\![0, n-1]\!] = [0, 1, \ldots, n-1]$
- 场上存活的玩家（无序）集合为：S，定义 S 中最大、最小编号分别为 $\overline{s}, \underline{s}$

- 在所有报数之后，最后的玩家编号为 w，即我们所求；定义函数 $f:\mathbb{R}\to\mathbb{R}$，输入总人数 n，输出最终玩家：

$$f:n\mapsto w$$

- 在一次报数后，“退出”的人编号为 x；
- 在一次报数、x 退出后，待抽签序列为 $P_{n-1}=[x+1,\dots,\overline{s},\underline{s},\dots,x-1]$.

    - > *这么定义是为了符合报数规则*

- 对 P 序列进行不断操作，定义函数 $g:\mathbb{R}\to\mathbb{R}$，为从拥有 n 个元素的 P 开始得到最终玩家胜利编号 w 的映射

$$g:n\mapsto w$$

- 设 L_1,L_2 为两个包含参数个数相同，且参数两两不同的两个序列，定义 $p:\mathbb{R}\to\mathbb{R}$ 为两者编码之间一一映射关系。举例：假设在 L_1,L_2 两次编码中，在 a,b 实际位置上两序列中的编码如下图所示：

实际位置	a	...	b
L_1	3	...	1
L_2	7	...	0

则，$p(3)=7,p(1)=0$

对从零开始标号序列操作

1. 起始数列

$$L_n=[0,1,\dots,n-1],\quad f(n)=w$$

2. 报数后求出被移除的数的表达式：$x=(2-1)\%n=1\%n$，一定要记得减 1，这是我们编码决定的。
3. 移除后，待报数序列如下图所示。易得，分别对 L_n 和 P_{n-1} 进行操作，两者得到的最终胜者应该相同（为 w）：

$$P_{n-1}=[x+1,\dots,n-1,0,1,\dots,x-1],\quad g(n-1)=w$$

4. 已知以下两个序列之间呈一一映射关系，我们假设该关系（从 P_{n-1} 到 L_{n-1}）为 $p:\mathbb{R}\to\mathbb{R}$. 重要的结论：**胜者的实际位置没有变化，变化的是其位置上的编码**。假设 L_{n-1} 得到的胜者为 $w':f(n-1)=w'$，在 P_{n-1} 中的胜者即为 $p^{-1}(w')$.

$$\begin{gathered}P_{n-1}:[x+1,\dots,n-1,0,1,\dots,x-1]\\ \downarrow p\\ L_{n-1}:[0,1,\dots,x-1,x+1,\dots,n-1]\end{gathered}$$

5. 建立关系

$$\begin{cases}f(n)=g(n-1)=w\\ \\ g(n-1)=p^{-1}(f(n-1))=p^{-1}(w')\end{cases}\Longrightarrow\boxed{f(n)=p^{-1}(f(n-1))}$$

6. 求函数 p^{-1}，两者成一次函数关系：

$$p^{-1}(t)=\begin{cases}t+(x+1),\quad 0<t+(x+1)<n\\ t+(x+1)-n,\quad t+(x+1)\geq n\end{cases}$$

   化简得到：

$$p^{-1}(t) = (t + x + 1)\%n$$

7. 代入式子：

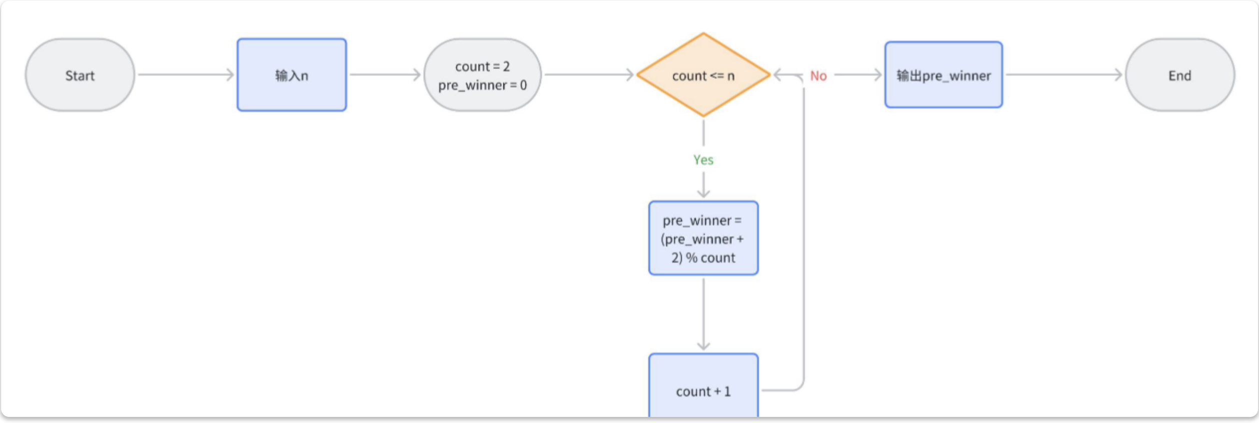
$$\begin{aligned} f(n) &= p^{-1}(f(n-1)) \\ &= (f(n-1) + x + 1)\%n \\ &= (f(n-1) + 1\%n + 1)\%n \\ &= (f(n-1) + 2)\%n \end{aligned}$$

结论：对 L_n 进行操作，最后剩余的玩家通过以下式子得到：

$$\boxed{f(n) = (f(n-1) + 2)\%n}$$

对任意序列任意操作

1. 当 $n = 1$ 时，易得 $f(n)|_{n=1} = 1$.
2. 当 $n > 1$ 时，求带有 n 玩家的问题，通过对上式不断递推得到。
所得结论即流程图所示图：



附录

A：源码

```
/* Homework : Josephus 问题
 * Author : Brandon Lin 林楠
 * Date : 25 September, 2023 */

#include <stdio.h>

// Declaration
int find_winner ( int player_amounts );

int main(){
  // Definitions (without initialization)
  int player_amounts; /* 玩家数量 */
  int pre_winner; /* 赢家 */

  // Get the amount of players
  printf("Insert number of players (n): ");
  scanf("%d", &player_amounts);

  // Find the final winner
  pre_winner = find_winner(player_amounts);
  // Print the result
  printf("In the case of %2d players, the winner is %d\n", player_amounts,
pre_winner);
  return 0;
}

int
find_winner ( int player_amounts )
{
  // Definitions
  int pre_winner = 0;
  int count;
  const int step = 2;

  // Run the game
  for ( count = step; count <= player_amounts; ++count ) {
    pre_winner = ( pre_winner + step ) % count;
  }

  return pre_winner;
}
```

B：数组实现

时间复杂度明显高于（效率更差）A 实现方式。
所以！数学理论才是王道。

```
/* Homework : Josephus 问题
 * Author : Brandon Lin 林楠
 * Date : 25 September, 2023 */

#include <stdio.h>

// Declaration
int find_winner(int player_amounts);

int main() {
    int player_amounts;
    int pre_winner;

    // 获取玩家数量
    printf("Insert number of players (n): ");
    scanf("%d", &player_amounts);

    // 找到最终赢家
    pre_winner = find_winner(player_amounts);

    // 打印最终结果
    printf("In the case of %2d players, the winner is %d\n", player_amounts,
pre_winner);
    return 0;
}

int find_winner(int player_amounts) {
    // 创建玩家列表
    int player_arr[player_amounts];
    int exist_players = player_amounts;
    for (int i = 0; i < player_amounts; i++) {
        player_arr[i] = i;
    }

    int current_player_index = -1; // 一定一定要从 -1 开始！！！！！！血的教训啊啊啊啊啊

    // 运行游戏
    while (exist_players > 1) {
        // 寻找下一个活着的玩家
        int count = 0;
        while (count < 2) {
            current_player_index = (current_player_index + 1) % player_amounts;
            if (player_arr[current_player_index] != -1) {
                count++;
            }
        }

        printf("current_player_index %d.\n", current_player_index);
        // 淘汰当前玩家
        player_arr[current_player_index] = -1;
        exist_players--;

    }

    // 返回赢家的编号
    for (int i = 0; i < player_amounts; i++) {
        if (player_arr[i] != -1) {
            return player_arr[i];
        }
    }

    return 0; // 只是不让编译器爆出警告……
}
```