西安交通大学实验报告

课程 **数据结构** 实验名称 **约瑟夫环** 共 **6** 页 实验报告日期 **2021**年**12**月**19**日

专业班级 **计算机006** 学号 **2206110705**  姓名 董宇轩

# **问题重述**

**设编号为1，2，…，n(n>0)个人按顺时针方向围坐一圈，每人持有一个正整数密码。开始时任意给出一个报数上限m，从第一个人开始顺时针方向自1起顺序报数，报到m时停止报数，报m的人出列，将他的密码作为新的m值，从他在顺时针方向上的下一个人起重新自1报数；如此下去直到所有人全部出列为止。**

# **分析过程**

S1:将每个人的密码值放入mat.txt文本文件中，导入该文件；

S2:输入第一次要报的数m；(不用输入total值，通过统计密码值的数来计算总人数；

S3:采用一个环形单链表（将末尾指针指到第一个节点）将所有的数据存入其中，环形单链表的结构包括要存放的元素elem，密码值key,标识符tag(用来判断当前人是否已经出局，初始时全置为0，若以出局则置为1,已经指向下一个节点的指针)

S4:从第一个人开始报数，设置一个计数器k=1，记录当前报数人数。

S5：如果当前报数的人所在节点的tag=0，k++；tag=1，不进行操作；然后指针往后移。

S6：重复上述S5。当k==m时，将目前指针所在节点的tag置为1，表明已出局，并将k置为1,开始新的一轮操作。

S7：重复S6，直到所有人都出局。每次出局时将人的标号elem写入num\_off.txt文本文件中。

# **源代码**

1. #define \_CRT\_SECURE\_NO\_WARNINGS
2. #include<stdio.h>
3. #include<stdlib.h>
4. #define Keytype int
5. #define Elemtype int
6. typedef struct Node {
7. Elemtype elem;
8. Keytype key;
9. int tag;
10. struct Node\* next;
11. }Node, \* ptr;*//ptr为指向结构体的指针*
12. void Input\_circle(ptr Head, int total,int mat[]);  *//向单链表中输入元素*
13. void Num\_Off(ptr Head, int total,int m,FILE\* fp);  *//约瑟夫环核心函数*
14. void Run\_process(void);
15. int main()
16. {
17. Run\_process();
18. return 0;
19. }
20. void Run\_process(void)
21. {
22. ptr my\_circle;
23. my\_circle = (ptr)malloc(sizeof(Node));
24. FILE\* fp;
25. FILE\* fp1;*//用来写入输出结果*
26. int mat[300];
27. int total = 0;
28. int m;*//报数上限*
29. printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*约瑟夫环问题仿真\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");
30. printf("请任意输入首次报数上限:");
31. scanf\_s("%d", &m);
32. *//构建约瑟夫环*
33. if ((fp = fopen("mat.txt", "r")) == NULL) {
34. perror("the file fail to read");
35. system("pause");
36. exit(1);
37. }
38. if ((fp1 = fopen("num\_off.txt", "wb+")) == NULL){
39. perror("the file fail to write");
40. system("pause");
41. exit(1);
42. }
43. fprintf(fp1, "初始数据如下:\n");
44. while (!feof(fp) && !ferror(fp)) {
45. fscanf(fp, "%d", &mat[total]);
46. fprintf(fp1, "%d ",mat[total]);
47. total++;
48. }
49. fprintf(fp1, "\n总人数为%d,第一次报数上限为%d\n结果如下:\n", total, m);
50. *//开始报数*
51. Input\_circle(my\_circle, total, mat);
52. Num\_Off(my\_circle, total, m, fp1);
53. fclose(fp);
54. fclose(fp1);
55. return;
56. }
57. void Input\_circle(ptr Head, int total,int mat[])
58. {
59. Head->tag = 0;
60. Head->elem = 1;
61. Head->key = mat[0];
62. ptr q = Head;
63. ptr p;
64. for (int i=1; i < total; i++) {
65. p = (ptr)malloc(sizeof(Node));
66. if (p == NULL) {
67. printf("申请空间失败！\n");
68. return;
69. }*//if*
70. q->next = p;
71. q = q->next;
72. q->tag = 0;
73. q->elem = i+1;
74. q->key = mat[i];
75. }*//for*
76. q->next = Head;*//组成环*
77. return;
78. }
79. void Num\_Off(ptr Head,int total, int m,FILE\* fp)
80. {
81. int count = 1;
82. int flag = m;
83. ptr p = Head;
84. while (count <= total) {
85. int k = 1;
86. int temp = flag;
87. while (k <= temp) {
88. if (p->tag == 0) {
89. if (k == temp) {
90. printf("%d ", p->elem);
91. fprintf(fp, "%d ", p->elem);
92. count++;
93. p->tag = 1;
94. flag= p->key;
95. }
96. p = p->next;
97. k++;
98. }
99. else p = p->next;
101. }
103. }
104. return;
105. }

# **测试**

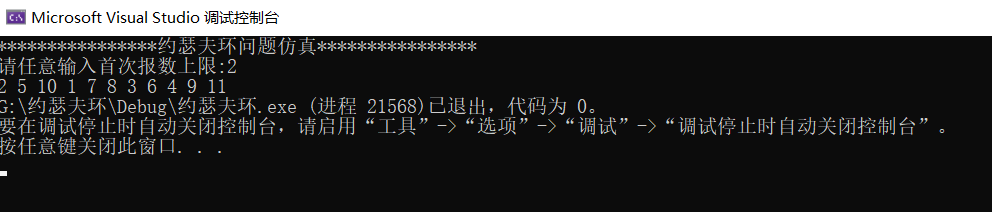
**第一组数据:**

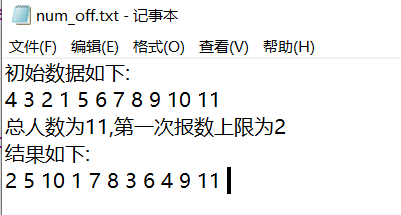
**4 3 2 1 5 6 7 8 9 10 11**

**总人数为11,第一次报数上限为2**

**理论结果如下:**

**2 5 10 1 7 8 3 6 4 9 11**





**第二组数据：**

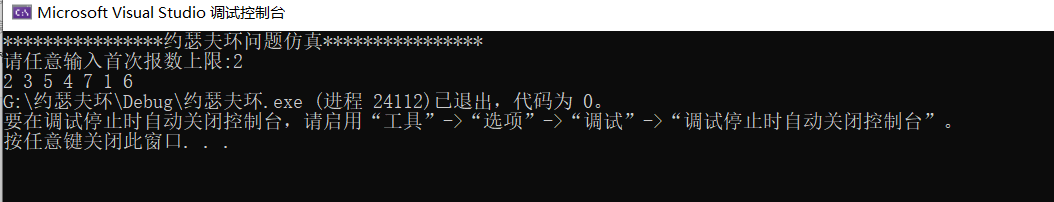
**初始数据如下:**

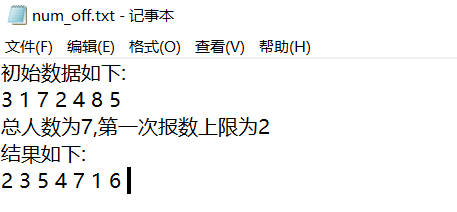
**3 1 7 2 4 8 5**

**总人数为7,第一次报数上限为2**

**理论结果如下:**

**2 3 5 4 7 1 6**





**第三组数据：**

**初始数据如下:**

**5 4 6 2 3 1 5 6 8 7 12 14 12 10 14 15 8 6 2 4 5 8 2 6 3**

**总人数为25,第一次报数上限为5**

**理论结果如下:**

**5 8 14 24 6 7 13 2 10 19 21 3 16 20 1 15 25 11 23 9 18 4 17 12 22**

