第 3 部分　课程设计项目

课程设计是 C 语言程序设计的一个重要环节，通过课程设计的综合训练，可以在学习理论知识的同时进一步提高解决实际问题的能力，强化综合应用能力，扩充知识，开阔视野。

本书课程设计部分共有 8 个项目，简单的项目可以一个人单独完成，复杂的项目可由几个人共同完成。每个项目都给出了目的与要求以及实现提示，希望读者可参考课程设计案例与提示实现课程设计。

项目 1　破解数字谜语

1. 目的与要求

有如下的数字谜语：

$$\begin{array}{r} \text{send} \\ +\ \text{more} \\ \hline \text{money} \end{array}$$

数字谜语的规则如下。

(1) 不同的英文字母表示不同的数字。

(2) 每一个数最左边一位不是 0，所以上面的谜题中，s、m、m 都不是 0。

(3) 上面的题目是加，那就表示 send 这个四位数与 more 这个四位数相加后，会得到 money 这个五位数，并且数字的安排如谜语所示。

试编写一个程序破解数字谜语。

2. 实现提示

s 与 m 最大值为 9，最大和为 9＋9＝18，因此进位最大为 1，由于数字谜语有进位，这样进位就为 1，也就是 m 取值 1。读者还可进步分析得出 o＝0。甚至得到更多的结果。

项目 2　利用计算机破案

1. 目的与要求

某处发生一案件，侦察结果得到如下可靠线索：

(1) 有 A、B、C、D 这 4 人有作案可能；

(2) A 和 B 中至少一人参与作案；

(3) B 和 C 中至少一人参与作案；

(4) C 和 D 中至少一人参与作案；

(5) A 和 C 中至少一人未参与作案；

(6) B 和 D 中至少一人未参与作案。

试编写一个程序进行破案。

2. 实现提示

用 0,1 表示 A、B、C、D 中某个是否参与作案(0 表示没有参与作案,1 表示参与作案)。后进行对 A、B、C、D 分别进行循环处理,并利用得到的线索判断筛选。

* 项目 3　任意阶魔方阵问题

1. 目的与要求

试解决任意阶魔方阵问题,n 阶魔方阵,就是把 $1\sim n^2$ 个连续的正整数填到一个 $n\times n$ 的方阵中,使得每列的和、每行的和以及两个对角线的和都相等。

2. 实现提示

案例 3 已解决了奇数阶魔方阵问题,现要求解决任意阶魔方阵问题,需要再解决偶数阶魔方阵问题即可,偶数阶魔方阵问题一般分为单偶数阶魔方阵问题和双偶数阶魔方阵问题,2(2k+1)阶魔方阵问题称为单偶阶魔方阵问题,4k 阶魔方阵问题称为双偶阶魔方阵问题,下面分别加以讨论。

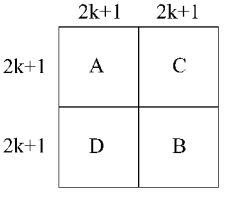

图 3.1　单偶阶魔方阵等分为 4 部分示意图

1) 单偶阶魔方阵问题

2(2k+1)阶魔方阵可以等分成 4 部分,每一个部分的长与宽都是 2k+1,如图 3.1 所示。

于是 A、B、C、D 这 4 个部分就都是奇数阶的。接着先填入一些准备用的数值：

(1) 在 A 中以奇数阶魔方阵的方式填入 $1\sim(2k+1)^2$。

(2) 在 B 中以奇数阶魔方阵的方式填入 $(2k+1)^2+1\sim2(2k+1)^2$。

(3) 在 C 中以奇数阶魔方阵的方式填入 $2(2k+1)^2+1\sim3(2k+1)^2$。

(4) 在 D 中以奇数阶魔方阵的方式填入 $3(2k+1)2+1\sim4(2k+1)^2$。

接下来对这 4 部分都要做一些修改才能达到魔方阵的境界,可以分成以下几个步骤来处理。

(5) 在 A 部分中间那一行的正中那一格起向左一共取 k 格,把它的内容与 D 的对应部分互换。

(6) 在 A 部分,除了中央行之外(中央行在上一步已经处理过了),每一行的头 k 个元素与 D 的对应部分互换。

(7) 在C部分，每一行的倒数k－1个元素(可能为0个)与B的对应部分互换。

做完上面步骤后就会得到一个2(2k＋1)阶的魔方阵了，对于k＝2，即2(2k＋1)＝10阶魔方阵为例，做完上面前4步，得到方阵如图3.2所示。

接着进行处理后3步，第5步是在A部分中央那一行正中央那一格起，向左取k＝2个格子出来，与D的对应部分互换，即13与32、6与81互换。第6步是把除了中央行外的其他行上前k个元素与D的对应部分互换，即17与92、24与99、23与98、5与80、10与85、12与87、11与86、18与93互换。最后的第7步在C部分，每一行的倒数m－1个元素(可能为0个)与B的对应部分互换，即40与65、41与66、47与72、28与53、34与59互换。最后的结果如图3.3所示。

17	24	1	8	15	67	74	51	58	65
23	5	7	14	16	73	55	57	64	66
4	6	13	20	22	54	56	63	70	72
10	12	19	21	3	60	62	69	71	53
11	18	25	2	9	61	68	75	52	59
92	99	76	83	90	42	49	26	33	40
98	80	82	89	91	48	30	32	39	41
79	81	88	95	97	29	31	38	45	47
85	87	94	96	78	35	37	44	46	28
86	93	100	77	84	36	43	50	27	34

图3.2　10阶单偶价魔方阵执行完前4步后的状态示意图

92	99	1	8	15	67	74	51	58	40
98	80	7	14	16	73	55	57	64	41
4	81	88	20	22	54	56	63	70	47
85	87	19	21	3	60	62	69	71	28
86	93	25	2	9	61	68	75	52	34
17	24	76	83	90	42	49	26	33	65
23	5	82	89	91	48	30	32	39	66
79	6	13	95	97	29	31	38	45	72
10	12	94	96	78	35	37	44	46	53
11	18	100	77	84	36	43	50	27	59

图3.3　10阶单偶价魔方阵执行完最后3步后的状态示意图

2) 双偶阶魔方阵问题

双偶阶魔方阵问题，也就是阶数是4的倍数(4k)，一定比读者想像的更简单，约略比奇数阶的复杂些而已。4k阶的魔方阵可以等分成4个部分，每一个部分的长与宽都是2k，如图3.4所示。

看$A_{2k\times 2k}$方阵。在这个方阵中的$4k^2$个元素中需要在某些元素上面做标号，方法是：在偶数号行(0,2,4,…)上的偶数列号都做上标记；但对奇数号行而言，则都在奇数号列上做标记。对于k＝2,8阶双偶阶魔方阵等分为4部分后A这一部分(＊就是记号)如图3.5所示。

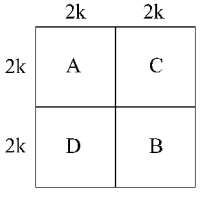

图3.4　双偶阶魔方阵等分为4部分示意图

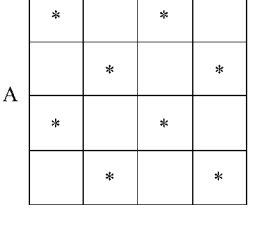

图3.5　双偶阶魔方阵等分为4部分后A部分的标号示意图

接着，将 A 的标号以 A 与 C 的分隔线为准，以对称的方式转到 C 部分，因此 A 与 C 就有了标号。得到的结果如图 3.6 所示。

实际上，对 A 部分，是对行号与列号之和为偶数的格子作标号，对 B 部分，是对行号与列号之和为奇数的格子作标号。

最后就是将 1～$(4k)^2$这些连续的整数填到方阵中。填数字的顺序是：自上而下一行一行地处理，在同一行上，则是从左向右一格一格地处理。数字的选择如下。

(1) 如果处理到的方格没有标号，令下一个值为 i，填在那里，接着找出那个以方阵中心为准与目前这一格对称的方格，把$(4k)^2-i+1$填到那里。

(2) 如果处理到的方格有标号，就找出那个以方阵中心为准的对称点，令下一个值为 i，将其填在那里，并且把$(4k)^2-i+1$填在目前的这一格上。

如果按照这个方法去填 8×8 的方阵，再以上述的标号为准，把 1～64 填到方阵中，会得到的 8 阶双偶阶魔方阵如图 3.7 所示。

*		*			*		*
	*		*	*		*	
*		*			*		*
	*		*	*		*	

图 3.6　双偶阶魔方阵标号示意图

64	2	62	4	5	59	7	57
9	55	11	53	52	14	50	16
48	18	46	20	21	43	23	41
25	39	27	37	36	30	34	32
33	31	35	29	28	38	26	40
24	42	22	44	45	19	47	17
49	15	51	13	12	54	10	56
8	58	6	60	61	3	63	1

图 3.7　8 阶双偶阶魔方阵示意图

*项目 4　制作万年历

1. 目的与要求

显示公元后任何年份的日历，日历以月份顺序排列，每月以星期顺序排列，类似于一般挂历上的格式，可参考如下格式：

```
输入年份:2010
                    2010 年
------------------------------------------------
                    一月
星期日  星期一  星期二  星期三  星期四  星期五  星期六
                                        1       2
  3       4       5       6       7       8       9
```

```
 10      11      12      13      14      15      16
 17      18      19      20      21      22      23
 24      25      26      27      28      29      30
 31
--------------------------------------------------
--------------------------------------------------
                    二月
星期日  星期一  星期二  星期三  星期四  星期五  星期六
         1       2       3       4       5       6
  7      8       9      10      11      12      13
 14      15      16      17      18      19      20
 21      22      23      24      25      26      27
 28
--------------------------------------------------
--------------------------------------------------
                    三月
星期日  星期一  星期二  星期三  星期四  星期五  星期六
         1       2       3       4       5       6
  7      8       9      10      11      12      13
 14      15      16      17      18      19      20
 21      22      23      24      25      26      27
 28      29      30      31
--------------------------------------------------
--------------------------------------------------
                    四月
星期日  星期一  星期二  星期三  星期四  星期五  星期六
                                 1       2       3
  4      5       6       7       8       9      10
 11      12      13      14      15      16      17
 18      19      20      21      22      23      24
 25      26      27      28      29      30
--------------------------------------------------
--------------------------------------------------
                    五月
星期日  星期一  星期二  星期三  星期四  星期五  星期六
                                                 1
  2      3       4       5       6       7       8
  9      10      11      12      13      14      15
 16      17      18      19      20      21      22
 23      24      25      26      27      28      29
 30      31
--------------------------------------------------
```

```
------------------------------------------------
                六月
星期日  星期一  星期二  星期三  星期四  星期五  星期六
                  1       2       3       4       5
  6       7       8       9      10      11      12
 13      14      15      16      17      18      19
 20      21      22      23      24      25      26
 27      28      29      30
------------------------------------------------
------------------------------------------------
                七月
星期日  星期一  星期二  星期三  星期四  星期五  星期六
                                  1       2       3
  4       5       6       7       8       9      10
 11      12      13      14      15      16      17
 18      19      20      21      22      23      24
 25      26      27      28      29      30      31
------------------------------------------------
------------------------------------------------
                八月
星期日  星期一  星期二  星期三  星期四  星期五  星期六
  1       2       3       4       5       6       7
  8       9      10      11      12      13      14
 15      16      17      18      19      20      21
 22      23      24      25      26      27      28
 29      30      31
------------------------------------------------
------------------------------------------------
                九月
星期日  星期一  星期二  星期三  星期四  星期五  星期六
                          1       2       3       4
  5       6       7       8       9      10      11
 12      13      14      15      16      17      18
 19      20      21      22      23      24      25
 26      27      28      29      30
------------------------------------------------
------------------------------------------------
                十月
星期日  星期一  星期二  星期三  星期四  星期五  星期六
                                          1       2
  3       4       5       6       7       8       9
 10      11      12      13      14      15      16
 17      18      19      20      21      22      23
 24      25      26      27      28      29      30
 31
```

```
------------------------------------------------
------------------------------------------------
                    十一月
星期日  星期一  星期二  星期三  星期四  星期五  星期六
                1       2       3       4       5       6
  7       8       9      10      11      12      13
 14      15      16      17      18      19      20
 21      22      23      24      25      26      27
 28      29      30
------------------------------------------------
------------------------------------------------
                    十二月
星期日  星期一  星期二  星期三  星期四  星期五  星期六
                        1       2       3       4
  5       6       7       8       9      10      11
 12      13      14      15      16      17      18
 19      20      21      22      23      24      25
 26      27      28      29      30      31
------------------------------------------------
请按任意键继续...
```

2. 实现提示

可按下面用伪代码描述的程序流程实现。

```
输入年份
显示年份
for (month=1; month<=12; month++)
{    /*打印每个月的日历*/
    打印一条横线
    显示月份
    显示"星期日 星期一 星期二 星期三 星期四 星期五 星期六"
    输出一个月的每一天
    打印一条横线
}
```

在显示日历时，要控制显示内容的位置，建议读者在需要时采用 printf()的格式串控制输出宽度，例如显示日历的年份可采用如下的代码：

```
printf("%25d年\n", year);              /*显示年份*/
```

对于上面代码中的“printf("%25d年\n", year)”中的“25”，读者可由实验确定。

**项目5　石头、剪刀、布游戏

1. 目的与要求

石头、剪刀、布游戏是一个传统的儿童游戏，在此游戏中，小孩子们用手表示石头、剪

刀或布中的一个，出拳头表示石头，伸出两根手指表示剪刀，伸手表示布，小孩子们面对面地数到三时做出他们的选择，如果所作选择是一样的，则表示平局，否则就按如下规则决定胜负：

(1) 石头砸坏剪刀；

(2) 剪刀剪碎布；

(3) 布覆盖石头。

2. 实现提示

为提高程序的可读性，可定义表示游戏结果的类型：

```
typedef enum
{    /*胜负结果：WIN(胜),LOSE(负)和 TIE(平)*/
   WIN, LOSE, TIE
} OutcomeType;
```

还可定义表示选择的类型如下：

```
typedef enum
{    /*选手可供的选择, ROCK(石头),SCISSOR(剪刀),CLOTH(布),DISPLAY(显示),
       HELP(帮助)和 QUIT(退出)*/
   ROCK, SCISSOR, CLOTH, DISPLAY, HELP, QUIT
} SelectType;
```

要玩游戏，机器和游戏者(用户)必须在布、石头和剪刀之间做出选择。可用函数SelectByMachine()与 SelectByPlayer()实现这些功能。在函数 SelectByMachine()实现中，通过用表达式 rand()%3 产生分布在 0～2 的随机整数来计算出机器的选择。因为该函数的类型是 SelectType，所以应把它的返回值转换成 SelectType 型。在函数SelectByPlayer()中，用 scanf("%c", &select)从输入流中读入数据，并把数据放在 select中。SelectByPlayer()返回的值依赖游戏者输入的信息。例如，如果游戏者输入字符 d，那么返回枚举值 DISPLAY。

一旦游戏者和机器都做出了选择，需要比较它们的选择，以决定游戏的结果。函数Compare()用于完成此项工作。

**项目 6　报数出列游戏

1. 目的与要求

m 个人按照顺时针方向坐成一圈，每人都有一个顺序号，顺序号依次为 1,2,3,…,m，开始时给定一个正整数 n，从第 1 个人开始按顺时针报数，让报到 n 的人出列，然后再从 1 开始报数，再让报到 n 的人出列，如此下去，直到所有人都出列为止。

2. 实现提示

报数出列游戏与约瑟夫问题类似，读者可采用解决约瑟夫问题的思路来解决问题，完全没有任何难度，为进一步提高读者的编程能力，下面提示采用递归函数解决报数出列游戏。

读者可仿照约瑟夫问题，采用如下的函数来求解报数出列游戏。

```
void NumberOff(int m, int n)
/*
    m个人围成一个圆圈，首先第1个人从1开始一个人一个人顺时针报数，报到
    第n个人，令其出列。然后再从下一个人开始，从1顺时针报数报到第n个人，
    再令其出列，…，如此下去，直到所有人都出圈为止
*/
```

由继续报数的条件是还有人没有出列，用 count 表示还没出列的人数，则报数的条件是“count＞0”，而函数 NumberOff()中无 count 参数，因此无法将函数 NumberOff()编写为递归函数，可采用辅助函数 NumberOffHelp()递归求解报数出列游戏，由于函数调用后参数 count 可能发生变化，因此采用指针作函数的参数，为方便起见，将相关的其他数据也作为参数，函数 NumberOffHelp()声明如下：

```
void NumberOffHelp(int m, int n, int * a, int * pPos, int * pNumber, int * pCount);
/*
    递归求解报数出列游戏
    m:人数
    n:最大报数数
    * pPos: 表示位置
    * pNumber:表示报数到的数
    * pCount:表示圈中剩下的人数
    a:指向存储m个人所围成的圈的数组
*/
```

可采用如下模式编写递归函数 NumberOffHelp()。

```
if (<递归调用条件>)
{    /*递归调用条件成立，继续进行递归调用*/
    递归调用方面的语句;
}
```

对于本课程设计项目，在本质上就是递归函数 NumberOffHelp()实现时将循环语句转换为 if 语句，只是要利用指针作函数的参数，大大增加程序难度，当然有利读者提高 C 语言的编程能力。

**项目7 武士巡逻问题

1. 目的与要求

在西洋棋中的武士(Knight)与象棋的马相似,走的是L形的路线(如图3.8所示),输入一个表示棋盘的大小值n,再输入一个起点的坐标,找出一条从那一个起点,可以让武士棋走完n^2格而不重复的路径。

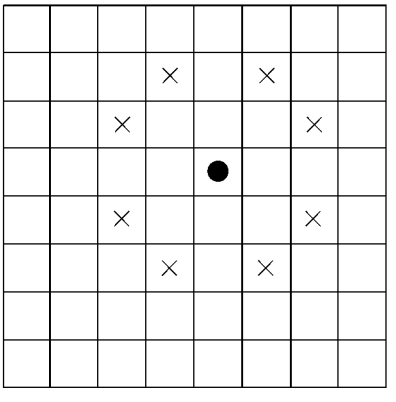

图3.8 武士走L形示意图(●表示当前位置,×表示下一可能走的位置)

2. 实现提示

首先看如何表示下一步可以走到的8个可能位置。如果目前位置的坐标为(x, y),那么8个位置就可以表示成(x+2, y+1)、(x+1, y+2)、(x−1, y+2)、(x−2, y+1)、(x−2, y−1)、(x−1, y−2)、(x+1, y−2)与(x+2, y−1),用xOffse[]与yOffset[]存放8个新位置与当前位置x与y坐标之差,xOffset[]={2, 1, −1, −2, −2, −1, 1, 2},yOffset[]={1, 2, 2, 1, −1, −2, −2, −1},则新坐标为(x+xOffset[i], y+yOffset[i])(i=0,1,…,7),用board[][]来存放棋盘,为方便起见,可不用0行0列,这样就需分配N+1行N+1列。

最简单的实现方式是仿照课程设计案例3的n皇后问题,采用回溯法递归求解。

回溯法递归将会搜索所有可能的解,会花费很多进间,当n=8,起始位置为(1, 1)时,在作者计算机上运行几个小时,只显示了一万多个解,并且还没运算完,因此建议得到一个解后就结束程序,也可改为找到能回到起始位置的解。

**项目8 员工工资管理系统

1. 目的与要求

设计一个利用文件处理方式实现对员工工资(包括员工编号、员工姓名、应发、扣款和实发)进行管理,具有增加数据、更新数据、查询数据、删除数据、列表显示数据以及重组文