



西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计 Programming in C



1011014

主讲：姜学锋，计算机学院

批量数据的遍历与访问

1、幻方编程

6.2.3 多维数组的引用



例题分析

双偶($n=4K$)阶魔方阵的填法。

(1) 先给出一个定义。互补：如果两个数字的和等于幻方最大数和最小数的和，即 $n*n+1$ ，称为互补。

(2) 先看看4阶幻方的填法：将数字从左到右、从上到下按顺序填写：

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16


6.2.3 多维数组的引用



例题分析

(3) 这个方阵的对角线，已经用颜色标出。将对角线上的数字，换成与它互补（同色）的数字。

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16



16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1

这里， $n*n+1 = 4*4+1 = 17$ ；把1换成 $17-1 = 16$ ；把6换成 $17-6 = 11$ ；把11换成 $17-11 = 6$换完后就是一个4阶幻方。

6.2.3 多维数组的引用



例题分析

双偶($n=4K$)阶魔方阵的填法。

- (1) 对于 $n=4k$ 阶幻方，先把数字按顺序填写。
- (2) 按 $4*4$ 把它划分成 $k*k$ 个方阵。因为 n 是4的倍数，一定能用 $4*4$ 的小方阵分割。
- (3) 然后把每个小方阵的对角线，像制作4阶幻方的方法一样，对角线上的数字换成互补的数字，就构成幻方。

6.2.3 多维数组的引用



例题分析

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

6.2.3 多维数组的引用

例6.56

```
1 #include <stdio.h>
2 #define N 100 //定义足够大的数组
3 int main()
4 { //求解4k阶魔方阵（双偶阶幻方）
5     int MA[N][N] = {0};
6     int i, j, m, n, L, S;
7     int x, y, ox, oy;
8     scanf("%d",&n); //输入n阶（n为4的倍数）
9     if (n>=4 && n<N && n%4==0) { //超过12列显示不下
10         x=y=0;
11         L=n*n;
12         for(i=1; i<=L; i++) { //先填数字
13             MA[y][x]=i;
14             if(i%n==0) x=0, y++;
15             else x++;
```

6.2.3 多维数组的引用

例6.56

```
1 #include <stdio.h>
2 #define N 100 //定义足够大的数组
3 int main()
4 { //求解4k阶魔方阵（双偶阶幻方）
5     int MA[N][N] = {0};
6     int i, j, m, n, L, S;
7     int x, y, ox, oy;
8     scanf("%d",&n); //输入n阶（n为4的倍数）
9     if (n>=4 && n<N && n%4==0) { //超过12列显示不下
10         x=y=0;
11         L=n*n;
12         for(i=1; i<=L; i++) { //先填数字
13             MA[y][x]=i;
14             if(i%n==0) x=0, y++;
15             else x++;
```


6.2.3 多维数组的引用

例6.56

```
16      }
17      S=1+L; //最大数和最小数之和
18      m=n/4; //以4x4大小分割魔方阵
19      x=y=ox=oy=0;
20      L=m*m;
21      for(j=1; j<=L; j++) {
22          for(i=0; i<4; i++) { //对每个4x4魔方阵做对角互补替换
23              MA[oy+i][ox+i]=S-MA[oy+i][ox+i];
24              MA[oy+i][ox+(4-i-1)]=S-MA[oy+i][ox+(4-i-1)];
25          }
26          if(j%m==0) ox=0, oy+=4;
27          else ox=j%m*4; //转移到下一个4x4魔方阵
28      }
29      for(i=0; i<n; i++) { //输出魔方阵
30          S=0; //S累计一行之和
```

6.2.3 多维数组的引用

例6.56

```
16     }
17     S=1+L; //最大数和最小数之和
18     m=n/4; //以4x4大小分割魔方阵
19     x=y=ox=oy=0;
20     L=m*m;
21     for(j=1; j<=L; j++) {
22         for(i=0; i<4; i++) { //对每个4x4魔方阵做对角互补替换
23             MA[oy+i][ox+i]=S-MA[oy+i][ox+i];
24             MA[oy+i][ox+(4-i-1)]=S-MA[oy+i][ox+(4-i-1)];
25         }
26         if(j%m==0) ox=0, oy+=4;
27         else ox=j%m*4; //转移到下一个4x4魔方阵
28     }
29     for(i=0; i<n; i++) { //输出魔方阵
30         S=0; //S累计一行之和
```

6.2.3 多维数组的引用

例6.56

```
31     printf(" ");
32     for(j=0; j<n; S=S+MA[i][j],j++)
33         printf("%4d ", MA[i][j]);
34     printf(" =%4d\n",S); //输出一行之和
35 }
36 printf("=");
37 for(j=0; j<n; j++) { //输出魔方阵列之和
38     S=0;
39     for(i=0; i<n; i++) S=S+MA[i][j];
40     printf("%4d ",S); //输出一列之和
41 }
42 L=S=0;
43 for(i=0; i<n; i++) S=S+MA[i][i] , L=L+MA[i][n-1-i];
44 printf(" =%4d =%4d\n",S,L); //输出对角线与反对角线之和
45 }
```

6.2.3 多维数组的引用

例6.56


```
46     else printf("error input: n=4k\n");  
47     return 0;  
48 }
```

若输入的n值不满足双偶($n=4K$)阶魔方阵，提示输入错误信息。

6.2.3 多维数组的引用

例6.56

程序运行屏幕



```
1 #include <stdio.h>
2 #define N 100
3 int main()
4 {
5     int a[N][8];
6     for (int i = 0; i < N; i++)
7     {
8         for (int j = 0; j < 8; j++)
9             a[i][j] = (i * 8 + j + 1) % 260;
10    }
```

```
64  2  3 61 60  6  7 57 = 260
 9 55 54 12 13 51 50 16 = 260
17 47 46 20 21 43 42 24 = 260
40 26 27 37 36 30 31 33 = 260
32 34 35 29 28 38 39 25 = 260
41 23 22 44 45 19 18 48 = 260
49 15 14 52 53 11 10 56 = 260
 8 58 59  5  4 62 63  1 = 260
= 260 260 260 260 260 260 260 260 = 260 = 260
```

8 ↙

6.2.3 多维数组的引用



例题分析

单偶($n=4K+2$)阶魔方阵的填法。

单偶阶魔方阵是三种之中最复杂的魔方阵。这里以 $n=10$ 为例。此时 $k=2$ 。

(1) 把方阵分为A、B、C、D四个象限，则每一个象限肯定是奇数阶。
用楼梯法，依次在A、D、B、C象限按奇数阶魔方阵的填法填数。

6.2.3 多维数组的引用



例题分析

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

6.2.3 多维数组的引用



例题分析

(2) 在A象限的中间行、中间格开始，按自左向右的方向，标出k格。A象限的其它行则标出最左边的k格。将这些格和C象限相应位置上的数互换位置。

6.2.3 多维数组的引用



例题分析

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

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

6.2.3 多维数组的引用



例题分析

(3) 在B象限任一行的中间格，自右向左标出 $k-1$ 列。（注意：当 $k-1 \leq 0$ 时，不用作此步骤），将B象限标出的这些数，和D象限相应位置上的数进行交换，就形成10阶幻方。

6.2.3 多维数组的引用



例题分析

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

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

6.2.3 多维数组的引用

例6.57

```
1 #include <stdio.h>
2 #define N 100 //定义足够大的数组
3 int main()
4 { //求解4k+2阶魔方阵（单偶阶幻方）
5     int A[N][N] = {0};
6     int i, j, k, m, n, L, S;
7     int x, y, ox, oy;
8     scanf("%d",&n); //输入n阶（n>=6, n=4k+2）
9     if (n>=6 && n<N && n%2==0 && n%4!=0) { //超过14显示不下
10         k=n/2;
11         for (j=0; j<4 ; j++) { //步骤1：依次填写A、D、B、C象限
12             m=k*k*j; //填写n/2（奇数）阶魔方阵
13             L=m+k*k;
14             x=k/2;
15             y=0;
```

6.2.3 多维数组的引用

例6.57

```
1  #include <stdio.h>
2  #define N 100 //定义足够大的数组
3  int main()
4  { //求解4k+2阶魔方阵（单偶阶幻方）
5      int A[N][N] = {0};
6      int i, j, k, m, n, L, S;
7      int x, y, ox, oy;
8      scanf("%d",&n); //输入n阶（n>=6, n=4k+2）
9      if (n>=6 && n<N && n%2==0 && n%4!=0) { //超过14显示不下
10         k=n/2;
11         for (j=0; j<4 ; j++) { //步骤1：依次填写A、D、B、C象限
12             m=k*k*j; //填写n/2（奇数）阶魔方阵
13             L=m+k*k;
14             x=k/2;
15             y=0;
```

6.2.3 多维数组的引用

例6.57

```
16      ox=(j==0||j==3) ? 0 : n/2;
17      oy=(j%2==0) ? 0 : n/2;
18      for(i=m+1; i<=L; i++) {
19          A[oy+y][ox+x]=i;
20          if(i%k==0) y++;
21          else x++,y--; //Hourse法
22          x=(x%k+k)%k;
23          y=(y%k+k)%k;
24      }
25  }
26  m=(n-2)/4; //对ADBC象限奇数阶魔方阵做处理
27  for(i=0;i<n/2;i++) {
28      for(j=0;j<m;j++) { //步骤2
29          k=(i==n/4)?n/4+j:j;
30          L=A[i][k], A[i][k]=A[i+n/2][k], A[i+n/2][k]=L;
```

6.2.3 多维数组的引用

例6.57

```
31      }
32      for(j=0; j<m-1; j++) { //步骤3
33          k=n/2+n/4+j;
34          L=A[i][k], A[i][k]=A[i+n/2][k], A[i+n/2][k]=L;
35      }
36  }
37  for(i=0; i<n; i++) { //输出魔方阵
38      S=0; //S累计一行之和
39      printf(" ");
40      for(j=0; j<n; S=S+A[i][j], j++)
41          printf("%4d ", A[i][j]);
42      printf(" =%4d\n", S); //输出一行之和
43  }
44  printf("=");
45  for(j=0; j<n; j++) { //输出魔方阵列之和
```


6.2.3 多维数组的引用

例6.57

```
31      }
32      for(j=0; j<m-1; j++) { //步骤3
33          k=n/2+n/4+j;
34          L=A[i][k], A[i][k]=A[i+n/2][k], A[i+n/2][k]=L;
35      }
36  }
37  for(i=0; i<n; i++) { //输出魔方阵
38      S=0; //S累计一行之和
39      printf(" ");
40      for(j=0; j<n; S=S+A[i][j], j++)
41          printf("%4d ", A[i][j]);
42      printf(" =%4d\n", S); //输出一行之和
43  }
44  printf("=");
45  for(j=0; j<n; j++) { //输出魔方阵列之和
```


6.2.3 多维数组的引用

例6.57

```
46      S=0;
47      for(i=0; i<n; i++) S=S+A[i][j];
48      printf("%4d ",S); //输出一列之和
49  }
50  L=S=0;
51  for(i=0; i<n; i++) S=S+A[i][i] , L=L+A[i][n-1-i];
52  printf("  =%4d =%4d\n",S,L); //输出对角线与反对角线之和
53  }
54  else printf("error input: n=4k+2)\n");
55  return 0;
56 }
```

6.2.3 多维数组的引用

例6.57


```
46      S=0;
47      for(i=0; i<n; i++) S=S+A[i][j];
48      printf("%4d ",S); //输出一列之和
49  }
50  L=S=0;
51  for(i=0; i<n; i++) S=S+A[i][i] , L=L+A[i][n-1-i];
52  printf("  =%4d  =%4d\n",S,L); //输出对角线与反对角线之和
53  }
54  else printf("error input: n=4k+2)\n");
55  return 0;
56 }
```

若输入的n值不满足单偶($n=4K+2$)阶魔方阵，提示输入错误信息。

6.2.3 多维数组的引用

例6.57

程序运行屏幕



```
1 #include <stdio.h>
2 #define N 100
3
4 int main()
5 {
6     int a[N][6];
7     int i, j;
8     for (i = 0; i < N; i++)
9     {
10         for (j = 0; j < 6; j++)
11             a[i][j] = (i * 6 + j + 1) % 37;
12     }
13     for (i = 0; i < 6; i++)
14     {
15         for (j = 0; j < N; j++)
16             printf("%4d", a[j][i]);
17         printf(" = 111\n");
18     }
19     return 0;
20 }
```

```
35  1  6 26 19 24 = 111
 3 32  7 21 23 25 = 111
31  9  2 22 27 20 = 111
 8 28 33 17 10 15 = 111
30  5 34 12 14 16 = 111
 4 36 29 13 18 11 = 111
= 111 111 111 111 111 111 = 111 = 111
```

```
6
```

CP 程序设计