西北工業大學
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、矩阵运算

6.2.3 多维数组的引用

- （2）矩阵应用
- 二维数组、三维数组经常用于数学的行列式、矩阵、立体几何等问题求解上。

【例6.4】

求矩阵A的转置矩阵 A^T

$$A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} \qquad A^T=\begin{bmatrix}1 & 4\\ 2 & 5\\ 3 & 6\end{bmatrix}$$

6.2.3 多维数组的引用

例6.4

```
#include<stdio.h>
int main()
{
  int A[2][3]={{1,2,3},{4,5,6}} ,AT[3][2], i, j;
  for (i=0; i<2; i++) //求矩阵A的转置
    for (j=0; j<3; j++) AT[j][i]=A[i][j];
  printf("A=\n");
  for (i=0; i<2; i++) { //输出矩阵A
    for (j=0; j<3; j++) printf("%d ",A[i][j]);
    printf("\n");
  }
  printf("AT=\n");
  for (i=0; i<3; i++) { //输出转置矩阵AT
    for (j=0; j<2; j++) printf("%d ",AT[i][j]);
    printf("\n");
```

6.2.3 多维数组的引用

例6.4

```
    }
    return 0;
}
```

6.2.3 多维数组的引用

例6.4

程序运行屏幕

```
A=
1 2 3
4 5 6
AT=
1 4
2 5
3 6
```

【例6.5】

已知A、B矩阵如下，求矩阵乘法AB。

$$A=\begin{bmatrix}3 & 2 & -1\\ 2 & -3 & 5\end{bmatrix},\ B=\begin{bmatrix}1 & 3\\ -5 & 4\\ 3 & 6\end{bmatrix}$$

例题分析

根据矩阵乘法的定义有：

$$C_{m\times n} = A_{m\times p} \times B_{p\times n}$$

$$C_{ij} = \sum_{k=1}^{p} A_{ik} B_{kj} (i = 1, 2, \cdots, m, \ \ j = 1, 2, \cdots, n)$$

这里

$$m = 2, n = 2, k = 3$$

6.2.3 多维数组的引用

例6.5

```c
#include <stdio.h>
int main()
{
  int A[2][3]={{3,2,-1},{2,-3,5}};
  int B[3][2]={{1,3},{-5,4},{3,6}};
  int C[2][2], i,j,k;
  for (i=0; i<2; i++) //求矩阵乘法
    for (j=0; j<2; j++) {
      C[i][j]=0;
      for (k=0; k<3; k++) C[i][j]=C[i][j]+A[i][k]*B[k][j];
    }
  printf("C=\n");
  for (i=0; i<2; i++) { //输出C矩阵
    for (j=0; j<2; j++) printf("%3d ",C[i][j]);
    printf("\n");
```

6.2.3 多维数组的引用

例6.5

```
    }
    return 0;
}
```

6.2.3 多维数组的引用

例6.5

```
1 #include <stdio.h>
```

```
C=
-10  11
 32  24
```

```
i][j]=C[i][j]+A[i][k]*B[k][j];
```

出C矩阵

```
tf("%3d ",C[i][j]);
```

【魔方阵举例】

魔方阵，又称为幻方。最早记载于我国公元前500年的春秋时期《大戴礼》中，这说明我国人民早在2500年前就已经知道了幻方的排列规律。而在国外，公元130年，希腊人塞翁才第一次提起幻方。我国不仅拥用幻方的发明权，而且是对幻方进行深入研究的国家。公元13世纪的数学家杨辉已经编制出3－10阶幻方，记载在他1275年写的《续古摘厅算法》一书中。在欧洲，直到574年，德国著名画家丢功才绘制出了完整的4阶幻方。

【魔方阵举例】

编程输出n阶魔方阵。

例题分析

数学上已经证明，对于n>2，n阶魔方阵都存在。目前填写魔方阵的方法，把魔方阵分成了三类：

（1）奇数(n=2K+1)阶魔方阵；

（2）双偶(n=4K)阶魔方阵；

（3）单偶(n=4K+2)阶魔方阵；

每类又有各种各样的填写方法。

6.2.3 多维数组的引用

例题分析

奇数(n=2K+1)阶魔方阵的经典填法是罗伯特法。步骤为：

（a）把1(或最小的数)放在第一行正中；

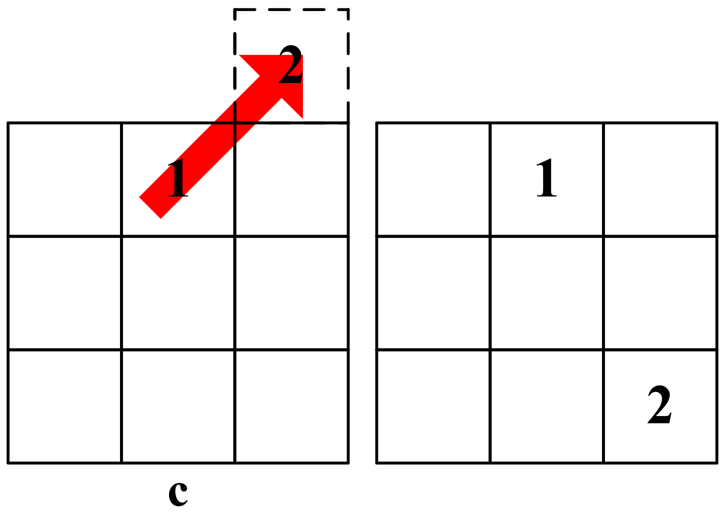
	1	

a

例题分析

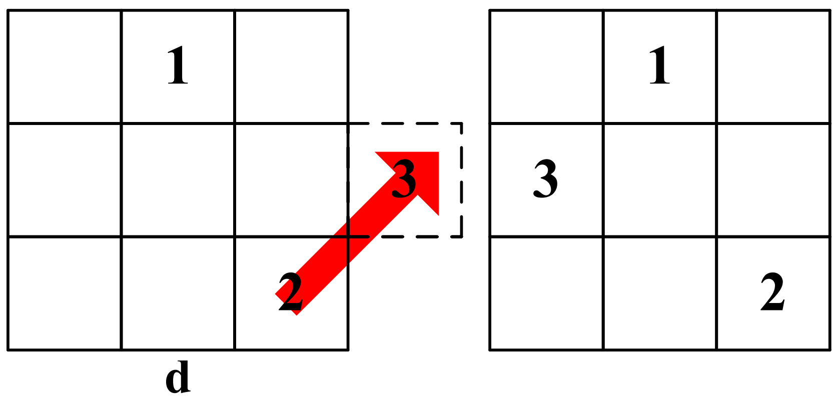
（b）每一个数放在前一个数的右上格；

（c）如果这个数所要放的格已经超出了顶行则把它放在底行，且仍然要放在右一列；



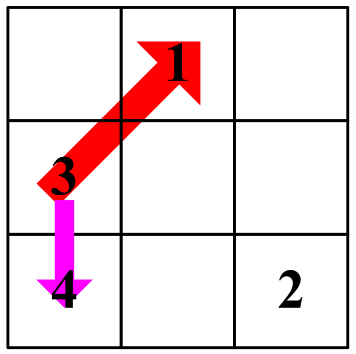
例题分析

（d）如果这个数所要放的格已经超出了最右列则把它放在最左列，且仍然要放在上一行；



例题分析

（e）如果这个数所要放的格已经有数填入，则把它放在前一个数的下一行同一列的格内；

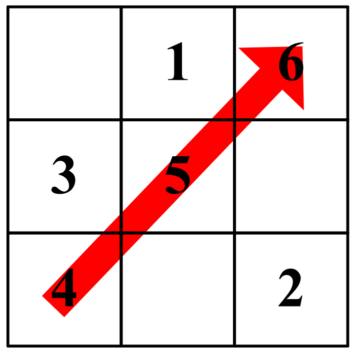


e

6.2.3 多维数组的引用

例题分析

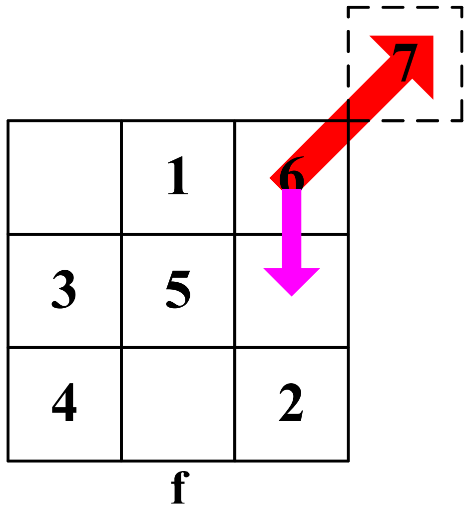
（b）每一个数放在前一个数的右上格；



b

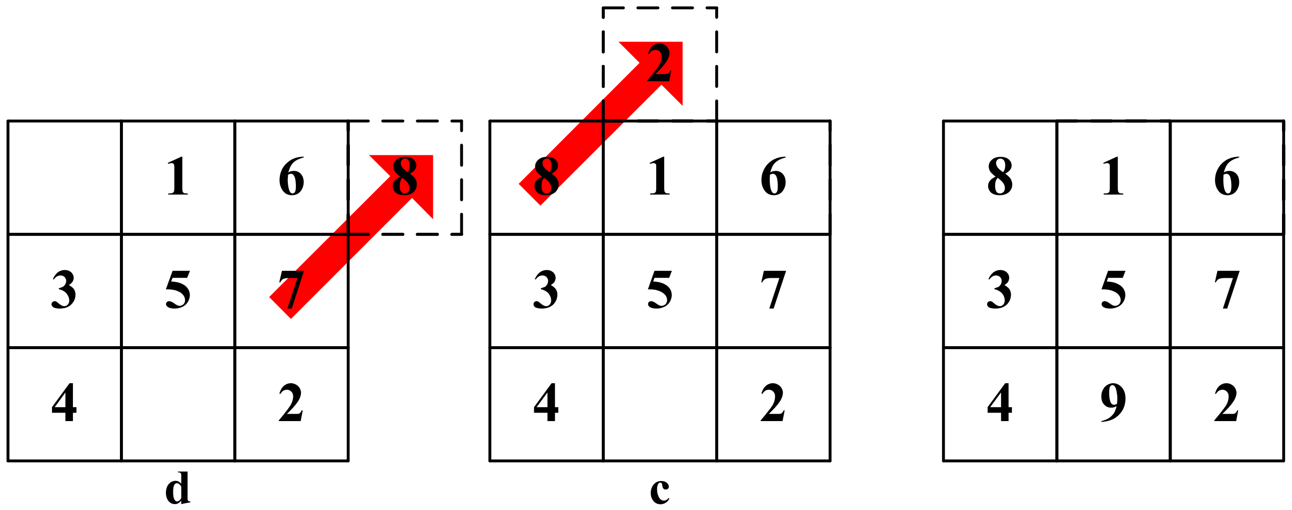
例题分析

（f）如果这个数所要放的格已经超出了顶行且超出了最右列，则把它放在前一个数的下一行同一列的格内。



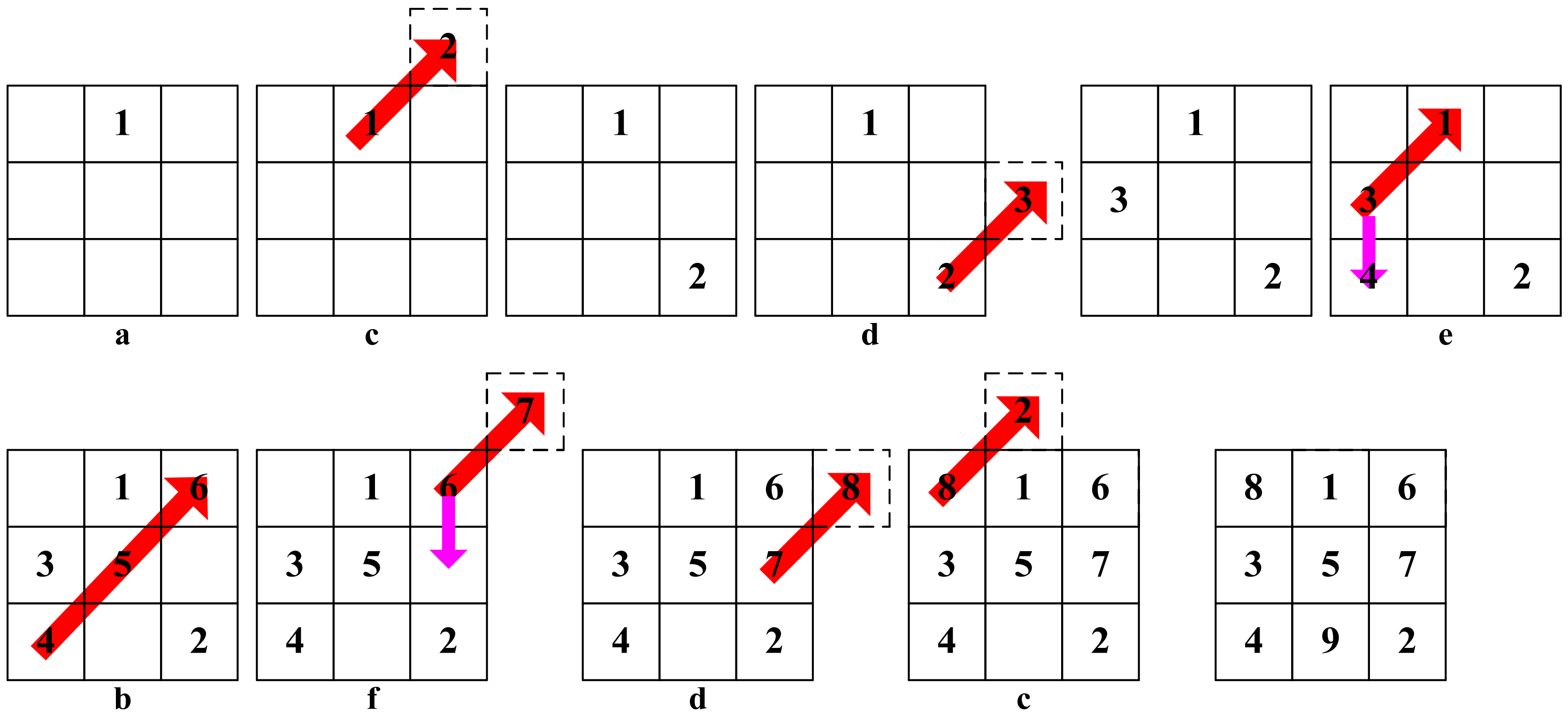
例题分析

这种写法总是先向“右上”的方向，像是在爬楼梯，故也称为楼梯法。



例题分析

奇数阶魔方阵的罗伯特法。



6.2.3 多维数组的引用

例6.55

```c
#include <stdio.h>
#define N 100 //定义足够大的数组
int main()
{ //求解2k+1阶魔方阵（奇数阶幻方）
  int MA[N][N] = {0};
  int i, j, m, n, L, S;
  int x, y, ox, oy;
  scanf("%d",&n); //输入n阶（n>=3，且为奇数）
  if (n>=3 && n<N && n%2==1) { //超过13列显示不下
    m=0; //从1开始填写n阶魔方阵
    L=m+n*n;
    x=n/2;
    ox=oy=y=0;
    for(i=m+1; i<=L; i++) {
      MA[oy+y][ox+x]=i;
```

6.2.3 多维数组的引用

例6.55

```
#include <stdio.h>
#define N 100 //定义足够大的数组
int main()
{ //求解2k+1阶魔方阵（奇数阶幻方）
  int MA[N][N] = {0};
  int i, j, m, n, L, S;
  int x, y, ox, oy;
  scanf("%d",&n); //输入n阶（n>=3，且为奇数）
  if (n>=3 && n<N && n%2==1) { //超过13列显示不下
    m=0; //从1开始填写n阶魔方阵
    L=m+n*n;
    x=n/2;
    ox=oy=y=0;
    for(i=m+1; i<=L; i++) {
      MA[oy+y][ox+x]=i;
```

6.2.3 多维数组的引用

例6.55

```
#include <stdio.h>
#define N 100 //定义足够大的数组
int main()
{ //求解2k+1阶魔方阵（奇数阶幻方）
  int MA[N][N] = {0};
  int i, j, m, n, L, S;
  int x, y, ox, oy;
  scanf("%d",&n); //输入n阶（n>=3，且为奇数）
  if (n>=3 && n<N && n%2==1) { //超过13列显示不下
    m=0; //从1开始填写n阶魔方阵
    L=m+n*n;
    x=n/2;
    ox=oy=y=0;
    for(i=m+1; i<=L; i++) {
      MA[oy+y][ox+x]=i;
```

6.2.3 多维数组的引用

例6.55

```
      if(i%n==0) y++;
      else x++,y--; //Hourse法
      x=(x%n+n)%n;
      y=(y%n+n)%n;
    }
    for(i=0; i<n; i++) { //输出魔方阵
      S=0; //S累计一行之和
      printf(" ");
      for(j=0; j<n; S=S+MA[i][j],j++)
        printf("%4d ", MA[i][j]);
      printf(" =%4d\n",S); //输出一行之和
    }
    printf("=");
    for(j=0; j<n; j++) { //输出魔方阵列之和
      S=0;
```

6.2.3 多维数组的引用

例6.55

```
            if(i%n==0) y++;
            else x++,y--; //Hourse法
            x=(x%n+n)%n;
            y=(y%n+n)%n;
        }
        for(i=0; i<n; i++) { //输出魔方阵
            S=0; //S累计一行之和
            printf(" ");
            for(j=0; j<n; S=S+MA[i][j],j++)
              printf("%4d ", MA[i][j]);
            printf(" =%4d\n",S); //输出一行之和
        }
        printf("=");
        for(j=0; j<n; j++) { //输出魔方阵列之和
            S=0;
```

6.2.3 多维数组的引用

例6.55

```
      if(i%n==0) y++;
      else x++,y--; //Hourse法
      x=(x%n+n)%n;
      y=(y%n+n)%n;
    }
    for(i=0; i<n; i++) { //输出魔方阵
      S=0; //S累计一行之和
      printf(" ");
      for(j=0; j<n; S=S+MA[i][j],j++)
        printf("%4d ", MA[i][j]);
      printf(" =%4d\n",S); //输出一行之和
    }
    printf("=");
    for(j=0; j<n; j++) { //输出魔方阵列之和
      S=0;
```

6.2.3 多维数组的引用

例6.55

```
            for(i=0; i<n; i++) S=S+MA[i][j];
            printf("%4d ",S); //输出一列之和
        }
        L=S=0;
        for(i=0; i<n; i++) S=S+MA[i][i] , L=L+MA[i][n-1-i];
        printf(" =%4d =%4d\n",S,L); //输出对角线与反对角线之和
    }
    else printf("error input: n=2k+1\n");
    return 0;
}
```

6.2.3 多维数组的引用

例6.55

```
          for(i=0; i<n; i++) S=S+MA[i][j];
          printf("%4d ",S); //输出一列之和
        }
        L=S=0;
        for(i=0; i<n; i++) S=S+MA[i][i] , L=L+MA[i][n-1-i];
        printf(" =%4d =%4d\n",S,L); //输出对角线与反对角线之和
    }
    else printf("error input: n=2k+1\n");
    return 0;
}
```

若输入的n值不满足奇数(n=2K+1)阶魔方阵，提示输入错误信息。

6.2.3 多维数组的引用

例6.55

```
#include <stdio.h>
```

程序运行屏幕

```
 17  24   1   8  15  =  65
 23   5   7  14  16  =  65
  4   6  13  20  22  =  65
 10  12  19  21   3  =  65
 11  18  25   2   9  =  65
=  65  65  65  65  65  =  65 =  65
```

```
5↙
```

程序设计