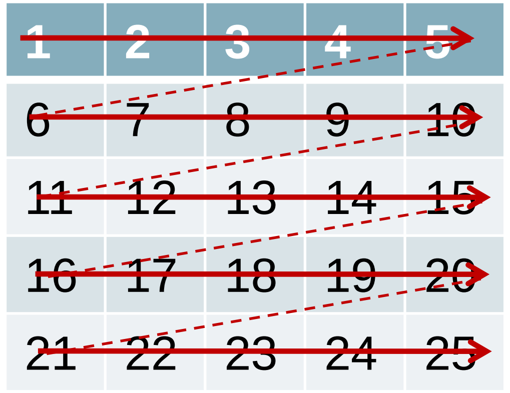
第7章 数组

——数组元素的访问与螺旋矩阵

数组元素的输入和输出

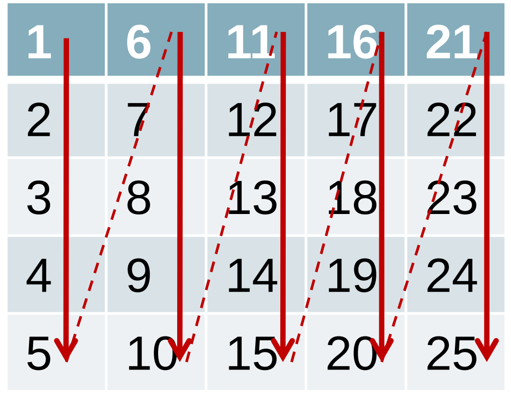


```
void SetArray(int a[][N], int m, int n)
{
    int i, j, len = 1;
    for (i=0; i<m; i++)
    {
          for (j=0; j<n; j++)
          {
              a[i][j] = len;
              len++;
          }
    }
}
```

```
void PrintArray(int a[][N], int m, int n)
{
    int i, j;
    for (i=0; i<m; i++)
    {
          for (j=0; j<n; j++)
          {
              printf("%d\t", a[i][j]);
          }
          printf("\n");
    }
}
```

数组元素的输入和输出

按列赋值



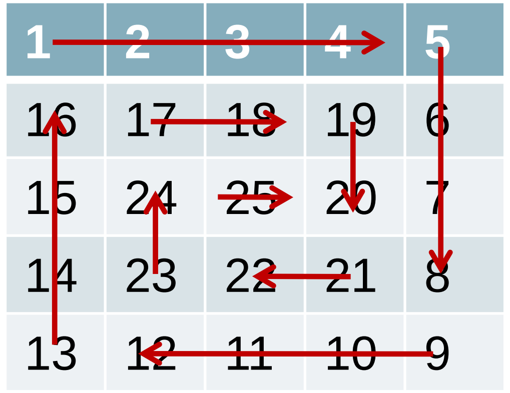
1	6	11	16	21
2	7	12	17	22
3	8	13	18	23
4	9	14	19	24
5	10	15	20	25

```c
void SetArray(int a[][N], int m, int n)
{
    int i, j, len = 1;
    for (j=0; j<n; j++)
    {
          for (i=0; i<m; i++)
          {
              a[i][j] = len;
              len++;
          }
    }
}
```

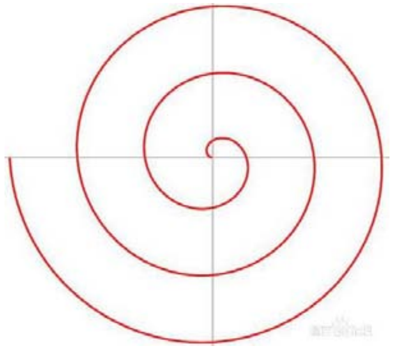
```c
void PrintArray(int a[][N], int m, int n)
{
    int i, j;
    for (i=0; i<m; i++)
    {
          for (j=0; j<n; j++)
          {
               printf("%d\t", a[i][j]);
          }
          printf("\n");
    }
}
```

数组元素的输入和输出

按圈赋值



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



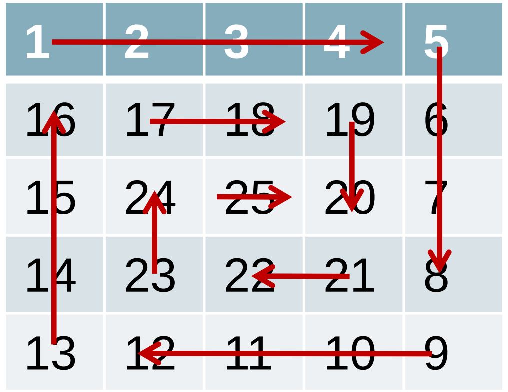
```
void SetArray(int a[][N], int len,
              int n)
{
   int m, k, level;
   level = n>0 ? (n+1)/2 : -1;
   for (m=0; m<level; m++)
   {
     //top
     //right
     //bottom
     //left
   }
}
```

螺旋矩阵

控制走过的圈数
(n+1)/2

螺旋矩阵

按圈赋值



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

```
void SetArray(int a[][N], int len,
              int n)
{
   int m, k, level;
   level = n>0 ? (n+1)/2 : -1;
   for (m=0; m<level; m++)
   {
     //top
     //right
     //bottom
     //left
   }
}
```

n=5 → level=3

第0圈，m=0

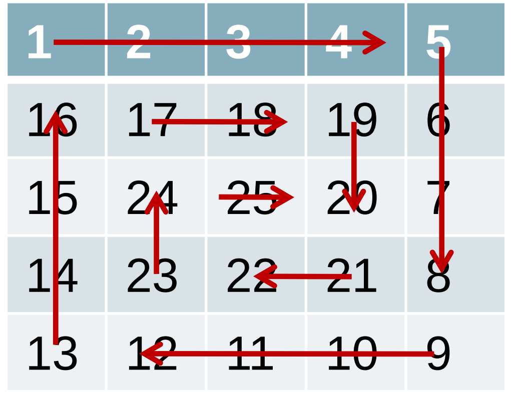
第1圈，m=1

第2圈，m=2

```
//top
for(k=m; k<n-m; k++)
{
    a[m][k] = len++;
}
//right
for(k=m+1; k<n-m-1; k++)
{
    a[k][n-m-1] = len++;
}
//bottom
for(k=n-m-1; k>m; k--)
{
    a[n-m-1][k] = len++;
}
//left
for(k=n-m-1; k>m; k--)
{
    a[k][m] = len++;
}
```

螺旋矩阵

按圈赋值



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

```
void SetArray(int a[][N], int len,
              int n)
{
   int m, k, level;
   level = n>0 ? (n+1)/2 : -1;
   for (m=0; m<level; m++)
   {
     //top
     //right
     //bottom
     //left
   }
}
```

```
#include<stdio.h>
#define N 10
void PrintArray(int a[][N], int m, int n);
void SetArray(int a[][N], int len, int n);
int main()
{
    int a[N][N], n;
    printf("Input n:");
    scanf("%d", &n);
    SetArray(a, 1, n);
    PrintArray(a, n, n);
    return 0;
}
```

```
//top
for(k=m; k<n-m; k++)
{
    a[m][k] = len++;
}
//right
for(k=m+1; k<n-m-1; k++)
{
    a[k][n-m-1] = len++;

n-1; k>m; k--)

n-1][k] = len++;

n-1; k>m; k--)

[m] = len++;
```

螺旋矩阵

迭代算法

```
SetArray(a, 1, n);
```

```
SetArray(a, 10, n);
```

```
void SetArray(int a[][N], int len,
              int n)
{
   int m, k, level;
   level = n>0 ? (n+1)/2 : -1;
   for (m=0; m<level; m++)
   {
     //top
     //right
     //bottom
     //left
   }
}
```

递归算法

```
SetArray(a, 1, n, 0);
```

```
SetArray(a, 10, n, 0);
```

```
void SetArray(int a[][N], int len,
              int n, int m)
{
   int k, level;
   level = n>0 ? (n+1)/2 : -1;
   if (m >= level) return;
   else
   {
     //top
     //right
     //bottom
     //left
     SetArray(a, len, n, m+1);
   }
}
```

螺旋矩阵

按圈赋值

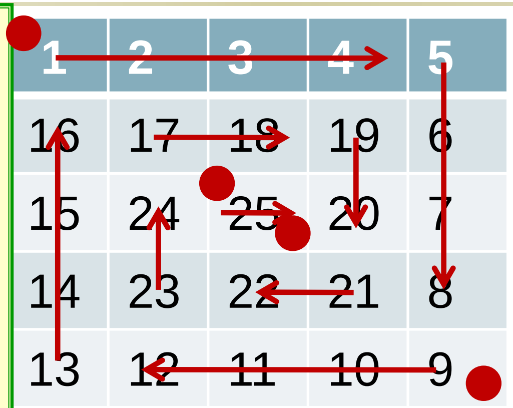
1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

控制走过的圈数
(n+1)/2

控制走过的格子数
n*n

螺旋矩阵

```
void SetArray(int a[][N], int len,
                 int n)
{
   int start=0, border=n-1, k, m=1;
   while (m <= n*n)
   {
     if (start > border) return;
     else if (start == border)
     {
         a[start][start] = len;
         return ;
     }
     else
     {
         //top
         //right
         //bottom
         //left
         start++;
         border--;
     }
   }
}
```



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

控制走过的圈数
(n+1)/2

⇩

控制走过的格子数
n*n

```
//top
for (k=start; k<=border-1; k++)
{
    a[start][k] = len++; m++;
}
//right
for (k=start; k<=border-1; k++)
{
    a[k][border] = len++; m++;
}
//bottom
for (k=border; k>=start+1; k--)
{
    a[border][k] = len++; m++;
}
//left
for (k=border; k>=start+1; k--)
{
    a[k][start] = len++; m++;
}
```

```
SetArray(a, 1, n);
```

```
SetArray(a, 10, n);
```

```
SetArray(a, 1, 0, n-1);
```

```
SetArray(a, 10, 0, n-1);
```

```
void SetArray(int a[][N], int len,
              int n)
{
   int start=0, border=n-1, k, m=1;
   while (m <= n*n)
   {
     if (start > border) return;
     else if (start == border)
     {
         a[start][start] = len;
         return ;
     }
     else
     {
         //top
         //right
         //bottom
         //left
         start++;
         border--;
     }
   }
}
```

控制走过的格子数
m <= n*n

```
void SetArray(int a[][N], int len,
              int start, int border)
{
  int k;
  while (start <= border)
  {
     if (start == border)
     {
         a[start][start] = len;
         return ;
     }
     else
     {
         //top
         //right
         //bottom
         //left
         start++;
         border--;
     }
  }
}
```

控制边界
start <= border

螺旋矩阵

```c
void SetArray(int a[][N], int len,
              int start, int border)
{
  int k;
  while (start <= border)
  {
     if (start == border)
     {
         a[start][start] = len;
         return ;
     }
     else
     {
         //top
         //right
         //bottom
         //left
         start++;
         border--;
     }
   }
}
```

迭代算法

```c
void SetArray(int a[][N], int len,
              int start, int border)
{
  int k;
  if (start > border) return;
  else if (start == border)
  {
    a[start][start] = len;
    return ;
  }
  else
  {
    //top
    //right
    //bottom
    //left
    SetArray(a, len, start+1, border-1);
  }
}
```

递归算法

讨论

- 1）将计算螺旋方阵的程序修改为计算螺旋矩阵，即行列数任意（不一定相等），程序如何修改？
- 2）按照下面的方向生成螺旋矩阵，程序如何修改？