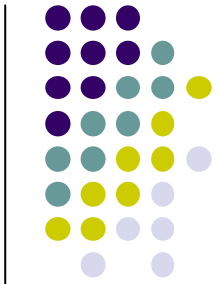
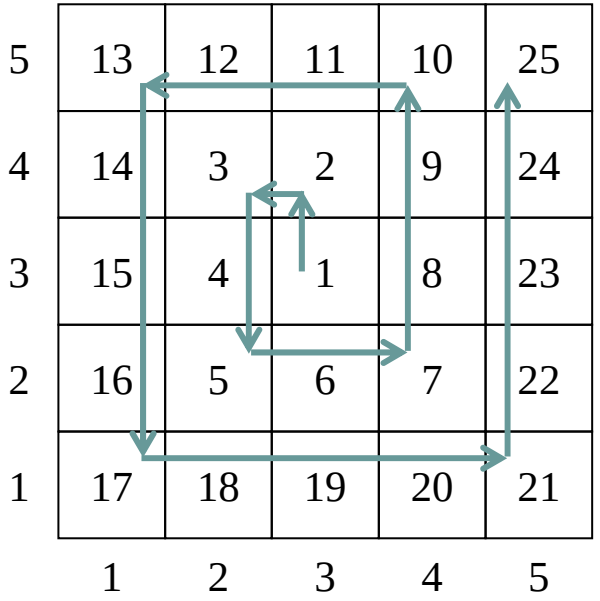


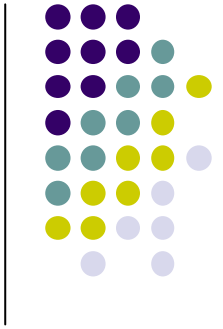
螺旋狀數字排列 (Spiral Tap) 程式裡的迴圈設計

問題

- 下圖的方格狀棋盤有 n 行、 n 列, n 爲奇數, 圖中橫座標和縱座標都是由 1 開始
- 由棋盤的正中央開始以螺旋狀方式逆時針順序排列 1 到 n^2 這些整數
- 請撰寫一個程式,
  輸入一整數 n 代表棋盤的寬度,
  輸入另一整數 t 代表目標數字,
  程式計算出 t 的橫座標與縱座標值



簡化與聯想



- 我們在課程裡還沒有正式介紹到陣列, 所以下面的說明裡我們都還是避開陣列的使用
  - 目標: 找到 target 對應的座標 (x, y)
  - 考慮右圖這個簡化的問題

y

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

x

簡單公式:

```
y = (target-1) / 5 + 1;   // “target” 在哪一列
x = (target-1) % 5 + 1; // “target” 在哪一行
```

  - 比較一般化的位置安排

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

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

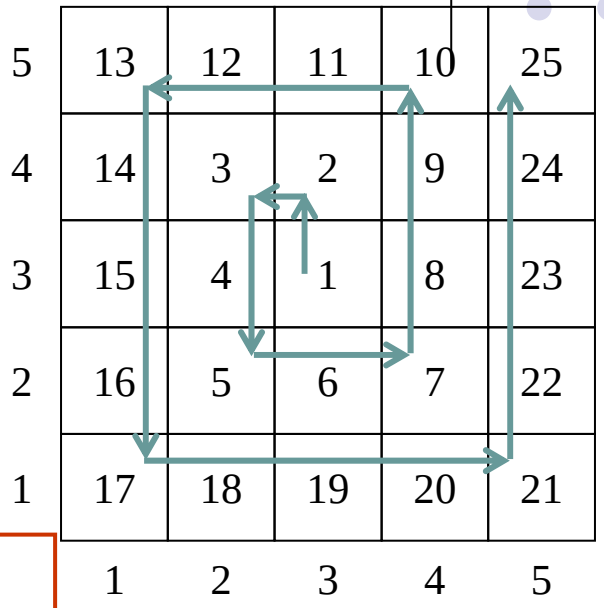
5	9	16	21	24	25
4	8	15	20	22	23
3	7	14	17	18	19
2	6	10	11	12	13
1	1	2	3	4	5
	1	2	3	4	5

  - 推導公式比較麻煩一些, 讓我們用『**模擬(Simulation)**』的方法來稍微暴力一點處理這個問題

```
for (i=1, x=y=1; i<target; i++) {
    x = x+1;
    if (x==6) x=1, y = y+1;
}
```

嘗試設計

- 基本方法: **模擬**
- **迴圈**: 重複做類似的事
- 在這個問題裡模擬安排每一個數字時看到**迴圈**了嗎?



尋找規律性

由 **1 (3,3)** 開始走 **target-1** 步

每一步 把數字加 **1**, 修改座標 **(x,y)**

- 如果是簡單的一維方格, 修改座標就容易了?

1	2	3	4	5	6	7

…

```
for (i=1,x=1; i<target; i++) x = x+1;
```

- 修改座標要依據方向, 每一步要注意是否轉向

版本一

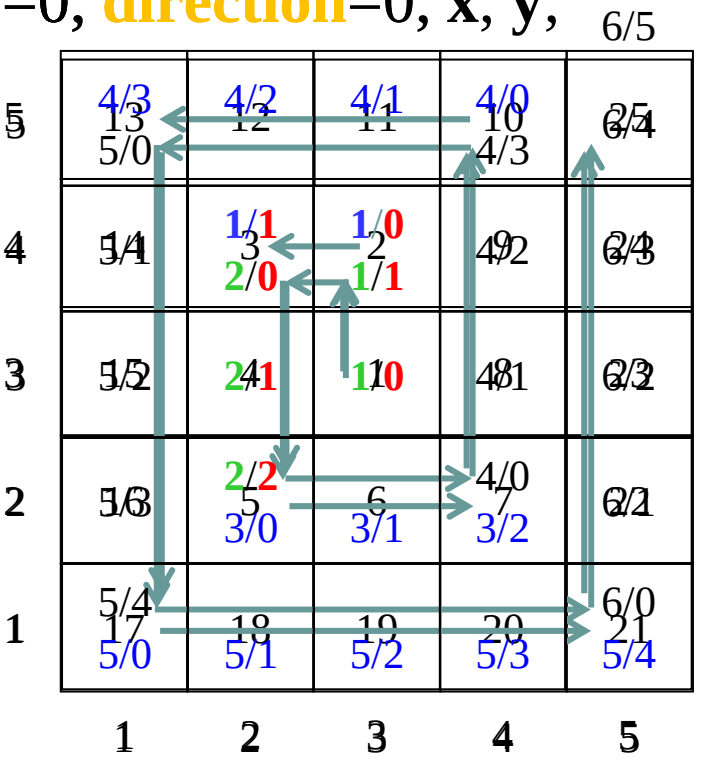
- **while** 迴圈直接由 **1, 2, 3, …** 開始往上計數
  每一直線段的長度 **1, 1, 2, 2, 3, 3, 4, 4, 5, …**

檢查是否 **target** <= n*n　　**path length** / **steps in each path**

```
int value=1; pathLen=1, step=0; nTurns=0; direction=0; x, y;
x = y = n/2 + 1;
while (value < target) {
    if (step == pathLen) {
        direction = (direction+1)%4;
        pathLen+=nTurns%2; nTurns++;
        step=0;
    }
    step++;
    value++;
    nextCoordinate(&x, &y, direction);
}
```

Directions:　**0:** 上　**1:** 左　**2:** 下　**3:** 右

下一次迴圈的資料



版本一 (續)

- void nextCoordinate(int *xPtr, int *yPtr, int direction)
  {
  switch (direction)
  {
  case 0: /* 上 */
  *yPtr = *yPtr + 1;
  break;
  case 1: /* 左 */
  *xPtr = *xPtr - 1;
  break;
  …
  }
  }

Directions:
0: 上
1: 左
2: 下
3: 右

還不熟悉 int * 的用法的話, 還是可以
把整個 switch 敘述放進前一頁的迴圈裡
取代 nextCoordinate(&x, &y, direction)

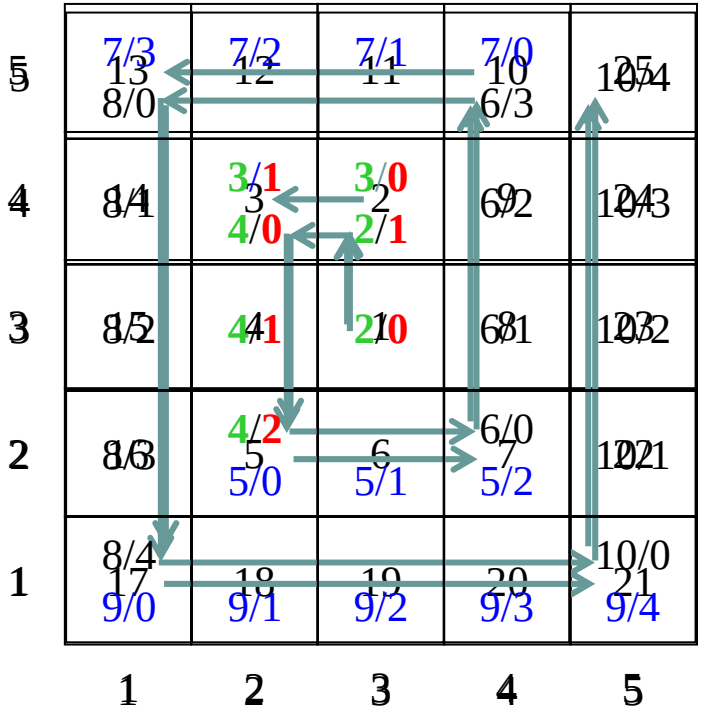
版本一 (簡化)

- **while** 迴圈直接由 **1, 2, 3, …** 開始往上計數
  每一直線段的長度 **1, 1, 2, 2, 3, 3, 4, 4, 5, …**

檢查是否 **target** <= n*n **÷2** **path length control** / **steps in each path**

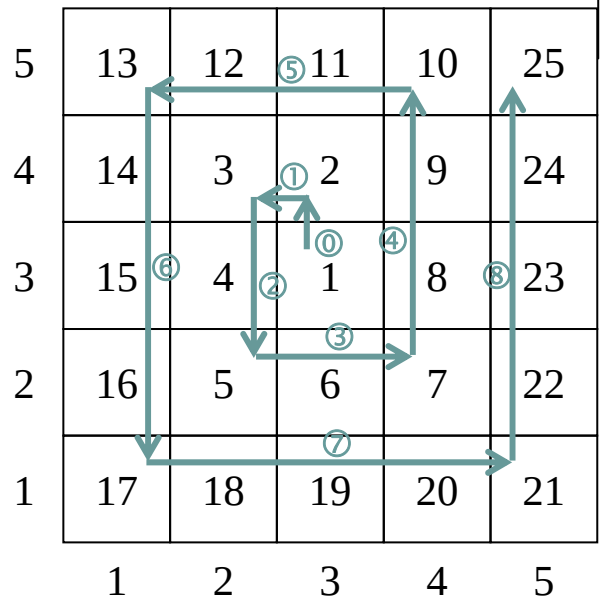
```
int count=0; pathLenCtrl=2, step=0; direction=0; x, y;
x = y = n/2 + 1;
while (++count < target)
{
    nextCoordinate(&x, &y, direction);
    if (++step == pathLenCtrl/2)
    {
        direction = (direction+1)%4;
        pathLenCtrl++, step=0;
    }
}
```

Directions: **0:** 上 **1:** 左 **2:** 下 **3:** 右



版本二: 計數迴圈 (counting loop)

- 分段的計數迴圈
  - n=5
  - 總共 2*n-1=9 直線線段
    ⓪, ①: 長度 1; ②, ③: 長度 2; …

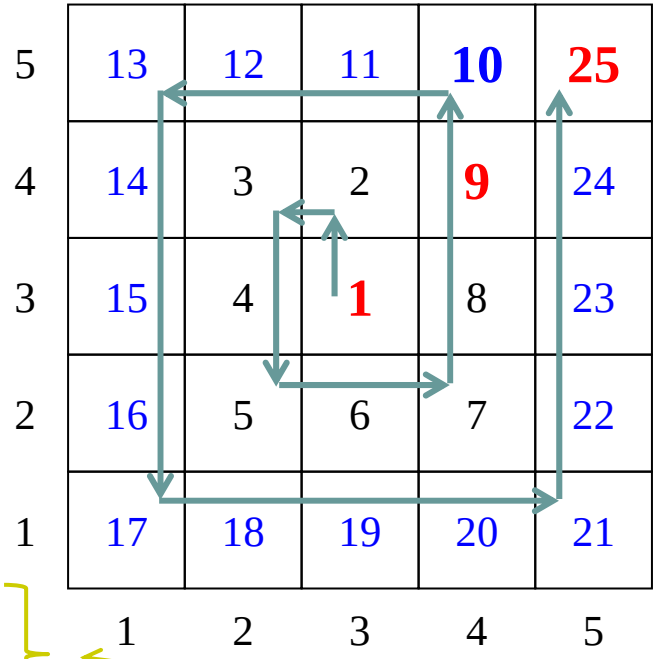



```
check if target <= n*n
value = 0;
x = y = n / 2 + 1;
for (i=0; i<2*n-1; i++)  // 第 i 個線段
    for (j=0; j<i/2+1; j++)  // 線段中第 j 步
    {
        if (++value == target) 輸出 x, y 並且結束兩層的迴圈
        nextCoordinate(&x, &y, i%4);
    }
```

方向: 0: 上 1: 左
2: 下 3: 右

版本三: 加快執行速度

- 最內層 k=0, 開始和結束的數字都是 (2k+1)(2k+1)=1
- 第二層 k=1, 開始和結束的數字分別是 (2k-1)(2k-1)+1=2 和 (2k+1)(2k+1)=9
- 第 k 層的開始數字是 **(2k-1)(2k-1)+1**
  結束數字是 **(2k+1)(2k+1)**
- 先用一個 for 迴圈找到滿足下式的 **k** 值
  (2k-1)(2k-1)<target<=(2k+1)(2k+1)
- count=(2*k-1)*(2*k-1); // 0~(k-1)層總數
  x = n/2+k+1, y=n/2+k+1; // 開始的座標
  for (i=1; i<=**4**; i++) { // 4 段直線
  for (j=0; j<**2*k**; j++) { // 每一段走 2k 步
  if (++count == target) 輸出 x, y 並且*結束*兩層的迴圈
  nextCoordinate(&x, &y, i%4);
  }
  }



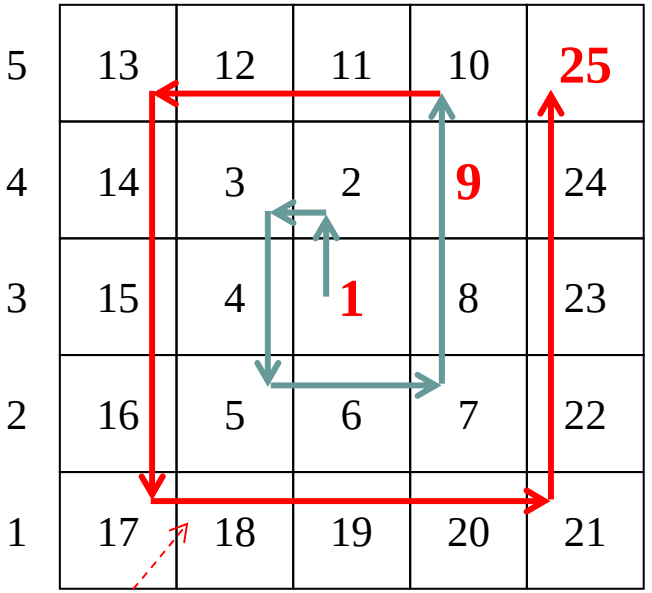
從 **10** 開始走時, 第一段少了一步, 所以假想 **9** 在 (5,5), 由 9 開始走

版本四: 直接計算公式

- 先用一個 for 迴圈找到滿足下式的 **k** 值
  (2k-1)(2k-1)<target<=(2k+1)(2k+1)
- infimum =(2*k-1)*(2*k-1);
  x = n/2+k, y=n/2+k;
- steps = target – infimum;
- 最外圈的四段:

0 1, 2, …, 2k,
1 2k+1, 2k+2, …, 4k,
2 4k+1, 4k+2, …, 6k,
3 6k+1, 6k+2, …, 8k

iseg = (steps-1) / (2*k);



- x-(steps-1) | x-(2k-1) | x-(2k-1)+(steps-4k) | x+1
  y+1 | y+1-(steps-2k) | y+1-2k | y+1-2k+(steps-6k)