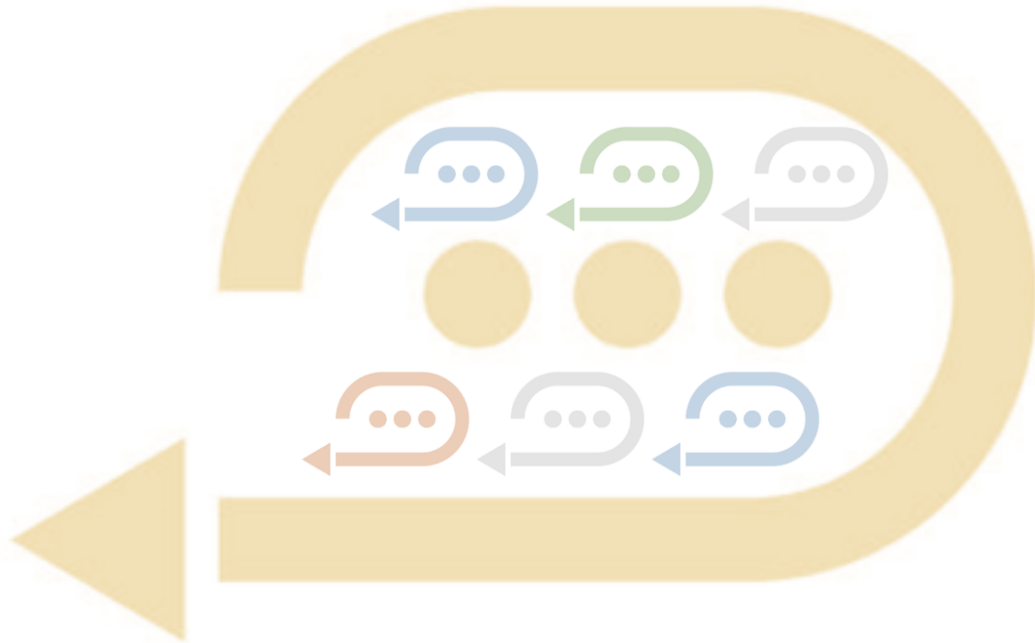




Computer Programming I: การเขียนโปรแกรมคอมพิวเตอร์ I

การซ้อน LOOP (Nested Loop)



อ.ดร.ปัญญนันท อ้นพงษ์

ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

aonpong_p@su.ac.th

Outline



- การซ้อน Loop
- แนวทางการประยุกต์ใช้การซ้อน Loop
- ตัวอย่างโจทย์

การซ้อน Loop



- หลังจากชั่วโมงนี้ไป เราจะเรียนรู้วิธีการเขียนโปรแกรมที่ซับซ้อนมากยิ่งขึ้น
- เริ่มต้นที่การซ้อน Loop
 - การซ้อน Loop คือการกำหนดการทำงานที่มีการวนซ้ำมากกว่า 1 ระดับ จะซ้อนกันกี่ชั้นก็ได้แล้วแต่สิ่งที่เราต้องการจะทำ

```
for(int j = 0; j < M; ++j) {  
    for(int i = 0; i < N; ++i) {  
        printf("%d %d", j, i);  
    }  
}
```

- ถ้ามีการซ้อน 2 ชั้นจะเรียกว่า Loop 2 ชั้น ถ้ามีการซ้อน 3 ชั้น จะเรียกว่า Loop 3 ชั้น
- โดยทั่วไปมักไม่ซ้อนเกินไปกว่า 3 ชั้น ถ้ามากเกินไปกว่านี้ให้พิจารณาโปรแกรมใหม่ ว่าเป็นสิ่งที่ต้องการจะทำได้จริง ๆ หรือไม่ และมีแนวทางอื่นจะแก้ไขปัญหานี้ได้ดีกว่าหรือไม่ (แม้ว่าบางครั้งจะเลี่ยงไม่ได้)

การซ้อน Loop: การทำงาน

- จริง ๆ แล้วการทำงานของ Loop ที่มีหลายชั้น ก็ไม่ได้มีความแตกต่างไปจากการทำงานคำสั่งธรรมดา
- ถ้าเราไล่อ่านโค้ดบรรทัดต่อบรรทัด การทำงานของมันก็เป็นไปตามลำดับขั้นตอนทั่วไป
- พิจารณาโปรแกรมนี้ ผลลัพธ์ของโปรแกรมคืออะไร

```
#include <stdio.h>
void main()
{
    int i, j;
    for(i = 0; i < 5; i++) {
        printf("*");
    }
}
```

การซ้อน Loop: การทำงาน

- ที่นี้เราลองครอบคำสั่ง loop for ที่เรามี ด้วย Loop for อีกชั้นหนึ่ง
- ลองเดาได้มั๊ยจะเกิดอะไรขึ้น

```
#include <stdio.h>
void main()
{
    int i, j;
    for(i = 0; i < 5; i++) {
        printf("*");
    }
}
```

```
#include <stdio.h>
void main()
{
    int i, j;
    for(j = 0; j < 3; j++) {
        for(i = 0; i < 5; i++) {
            printf("*");
        }
    }
}
```

การซ้อน Loop: การทำงาน

- สิ่งที่เกิดขึ้นก็คือ โปรแกรมจะทำงานที่อยู่ในกรอบสีแดงซ้ำ 3 รอบ ตามที่กำหนด

```
#include <stdio.h>
void main()
{
    int i, j;
    for(i = 0; i < 5; i++) {
        printf("*");
    }
}
```

```
#include <stdio.h>
void main()
{
    int i, j;
    for(j = 0; j < 3; j++) {
        for(i = 0; i < 5; i++) {
            printf("*");
        }
    }
}
```

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

การซ้อน Loop: การทำงาน

- สิ่งที่เกิดขึ้นก็คือ โปรแกรมจะทำงานที่อยู่ในกรอบสีแดงซ้ำ 3 รอบ ตามที่กำหนด

```
#include <stdio.h>
void main()
{
    int i, j;

    for (i=0; i<5; i++){
        printf("*");
    }

    for (i=0; i<5; i++){
        printf("*");
    }

    for (i=0; i<5; i++){
        printf("*");
    }
}
```

ทำซ้ำครั้งที่ 1

ทำซ้ำครั้งที่ 2

ทำซ้ำครั้งที่ 3

```
#include <stdio.h>
void main()
{
    int i, j;
    for(j = 0; j < 3; j++) {
        for(i = 0; i < 5; i++) {
            printf("*");
        }
    }
}
```

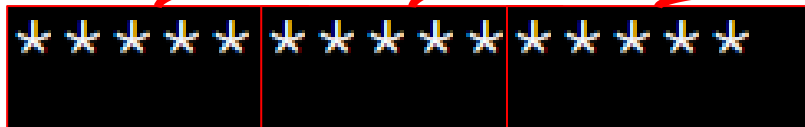


การซ้อน Loop: การทำงาน

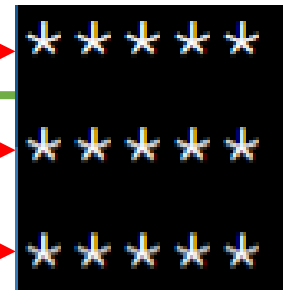
- คำสั่งไม่ได้กำหนดตายตัวว่าต้องอยู่ตรงไหน ลองกำหนดให้ขึ้นบรรทัดใหม่ใน Loop นอกดู

```
#include <stdio.h>
void main()
{
    int i, j;
    for(j = 0; j < 3; j++) {
        for(i = 0; i < 5; i++) {
            printf("*");
        }
    }
}
```

```
#include <stdio.h>
void main()
{
    int i, j;
    for(j = 0; j < 3; j++) {
        for(i = 0; i < 5; i++) {
            printf("*");
        }
        printf("\n");
    }
}
```



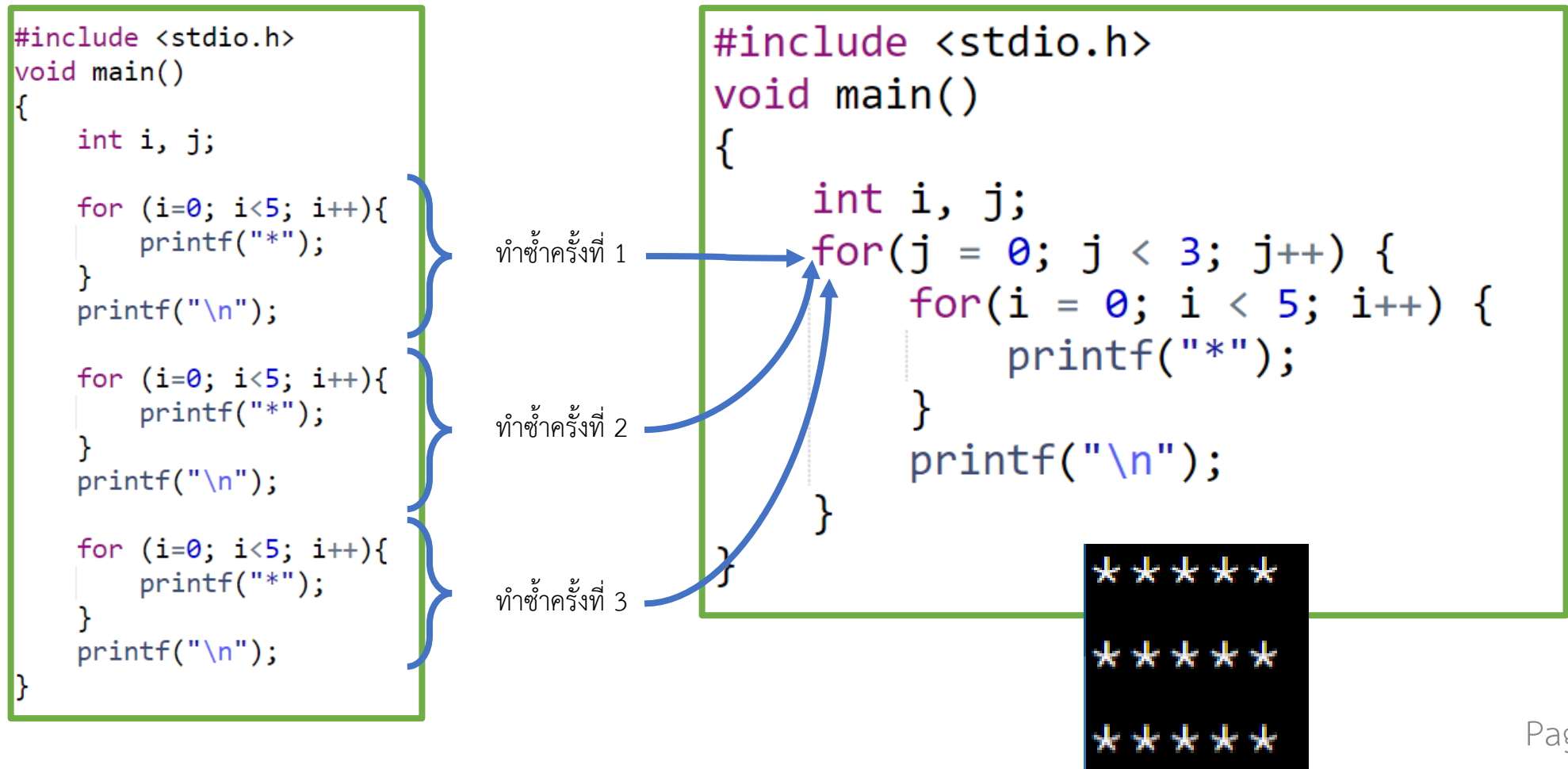
```
*****
*****
*****
```



```
*****
*****
*****
```


การซ้อน Loop: การทำงาน

- ถ้านำการซ้อน loop มาเปรียบเทียบกับแนวทางการเขียนแบบไม่ซ้อน loop ก็จะได้ประมาณนี้



การซ้อน Loop: การทำงาน

- เอาค่าของตัวแปรจาก loop นอกไปใช้ใน loop ในก็ได้เหมือนกัน

```
#include <stdio.h>
void main()
{
    int i, j;
    for(j = 0; j < 5; j++) {
        for(i = 0; i <= j; i++) {
            printf("*");
        }
        printf("\n");
    }
}
```



```
*
**
***
****
*****
```

การซ้อน Loop: การทำงาน

- สรุปได้ว่า การทำงานของ loop 2 ชั้น (หรือ 3, 4, ..) มีหลักการทำงานเหมือนกับ loop ชั้นเดียวทุกประการ ถ้าเข้าใจ loop ชั้นเดียวมาเป็นอย่างดี ก็สามารถขยับชั้นมาไม่ยาก
- แต่ความยากคือการประยุกต์ใช้ เพราะเมื่อเรารู้เรื่อง loop ที่ซ้อนกันได้ ความเป็นไปได้ที่จะสร้างโปรแกรมรูปแบบใหม่ ๆ ก็เพิ่มขึ้นมาก ทำให้โจทย์ที่จะได้เจอมีความหลากหลายด้วย
- ก่อนหน้านี้เราเห็นตัวอย่างการแสดงผลเครื่องหมาย * ง่ายๆ กันไปแล้ว มาดูสิ่งที่ประยุกต์มากขึ้นกัน

Outline



- การซ้อน Loop
- แนวทางการประยุกต์ใช้การซ้อน Loop
 - การจัดการข้อมูลในรูปแบบตารางหรือเมทริกซ์
 - การจัดการรูปภาพ
- ตัวอย่างโจทย์

แนวทางการประยุกต์ใช้การซ้อน Loop: ตาราง

- พิจารณาโปรแกรมต่อไปนี้ โปรแกรมจะพิมพ์อะไรออกมา (ไม่มี loop)

```
#include <stdio.h>
void main()
{
    int i, j, row=0, col=0;
    printf("(%d, %d)", row, col);
}
```

แนวทางการประยุกต์ใช้การซ้อน Loop: ตาราง

- พิจารณาโปรแกรมต่อไปนี้ โปรแกรมจะพิมพ์อะไรออกมา (ไม่มี loop)

```
#include <stdio.h>
void main()
{
    int i, j, row=0, col=0;
    printf("(%d, %d)", row, col);
}
```

(0, 0)

แนวทางการประยุกต์ใช้การซ้อน Loop: ตาราง

- พิจารณาโปรแกรมต่อไปนี้ โปรแกรมจะพิมพ์อะไรออกมา (มี loop 1 ชั้น)

```
#include <stdio.h>
void main()
{
    int i=0, j=0, row=5, col=6;
    for (i=0; i<col; i++){
        printf("(%d, %d) ", j, i);
    }
}
```

แนวทางการประยุกต์ใช้การซ้อน Loop: ตาราง

- พิจารณาโปรแกรมต่อไปนี้ โปรแกรมจะพิมพ์อะไรออกมา (มี loop 1 ชั้น)

```
#include <stdio.h>
void main()
{
    int i=0, j=0, row=5, col=6;
    for (i=0; i<col; i++){
        printf("(%d, %d) ", j, i);
    }
}
```

(0, 0) (0, 1) (0, 2) (0, 3) (0, 4) (0, 5)

แนวทางการประยุกต์ใช้การซ้อน Loop: ตาราง

- พิจารณาโปรแกรมต่อไปนี้ โปรแกรมจะพิมพ์อะไรออกมา (มี loop 2 ชั้น)

```
#include <stdio.h>
void main()
{
    int i, j, row=5, col=6;
    for (j=0; j<row; j++){
        for (i=0; i<col; i++){
            printf("(%d, %d) ", j, i);
        }
        printf("\n");
    }
}
```

แนวทางการประยุกต์ใช้การซ้อน Loop: ตาราง

- พิจารณาโปรแกรมต่อไปนี้ โปรแกรมจะพิมพ์อะไรออกมา (มี loop 2 ชั้น)

```
#include <stdio.h>
void main()
{
    int i, j, row=5, col=6;
    for (j=0; j<row; j++){
        for (i=0; i<col; i++){
            printf("(%d, %d) ", j, i);
        }
        printf("\n");
    }
}
```

```
(0, 0) (0, 1) (0, 2) (0, 3) (0, 4) (0, 5)
(1, 0) (1, 1) (1, 2) (1, 3) (1, 4) (1, 5)
(2, 0) (2, 1) (2, 2) (2, 3) (2, 4) (2, 5)
(3, 0) (3, 1) (3, 2) (3, 3) (3, 4) (3, 5)
(4, 0) (4, 1) (4, 2) (4, 3) (4, 4) (4, 5)
```

แนวทางการประยุกต์ใช้การซ้อน Loop: ตาราง

- พิจารณาโปรแกรมต่อไปนี้ โปรแกรมจะพิมพ์อะไรออกมา (มี loop 2 ชั้น)

```
#include <stdio.h>
void main()
{
    int i, j, row=5, col=6;
    for (j=0; j<row; j++){
        for (i=0; i<col; i++){
            printf("(%d, %d) ", j, i);
        }
        printf("\n");
    }
}
```

เนื่องจากผลลัพธ์ถูกจัดการตามแถวก่อน เราจึงนำแถวออกมาเป็นลูปด้านนอก ส่วนคอลัมน์เป็น loop ด้านใน

Loop ด้านในพิมพ์ข้อความออกมา loop ด้านในจะต้องวนจนครบ 6 รอบก่อน จึงจะหลุดออกมา และใน 6 รอบนี้ค่า j จะเหมือนเดิมเพราะลูปด้านนอกไม่ถูกแตะต้อง ค่า j จึงไม่เปลี่ยน

เมื่อออกจาก loop ด้านในแล้วจึงทำคำสั่งที่อยู่ถัดมา คือการขึ้นบรรทัดใหม่ จากนั้นวนกลับไปต้น loop ด้านนอก เพราะไม่มีคำสั่งอื่นให้ทำแล้ว (กลับไปตรวจสอบเงื่อนไขใหม่เหมือน loop ทั่วไป)

แนวทางการประยุกต์ใช้การซ้อน Loop: ตาราง

- **ลองคิดเพิ่มเติม** ถ้าเราอยากกำหนดให้สามารถปรับขนาดได้ตาม input จะแก้ไขยังไง (ให้รับค่า row และ col เข้ามาทางคีย์บอร์ด)

```
#include <stdio.h>
void main()
{
    int i, j, row=5, col=6;
    for (j=0; j<row; j++){
        for (i=0; i<col; i++){
            printf("(%d, %d) ", j, i);
        }
        printf("\n");
    }
}
```

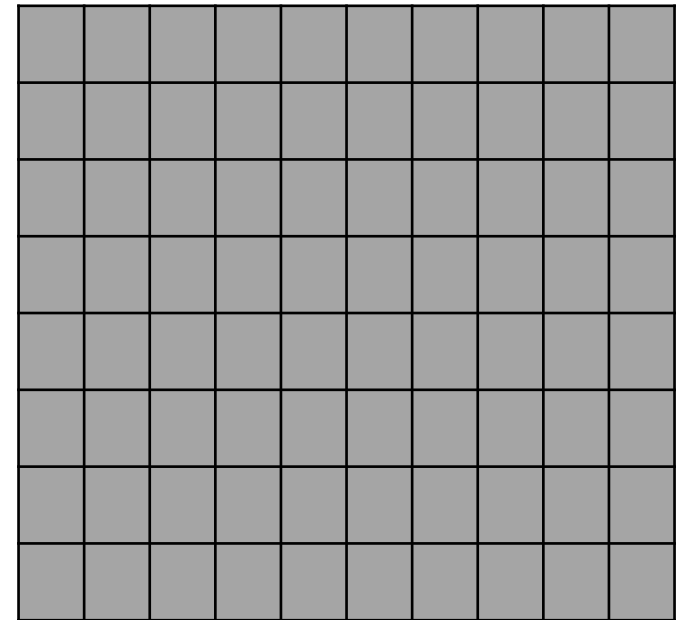
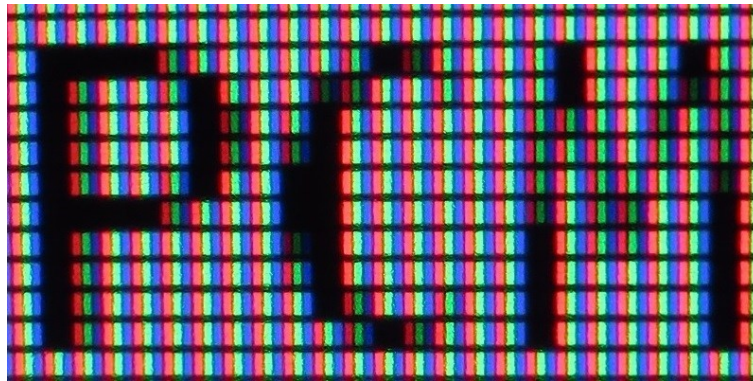
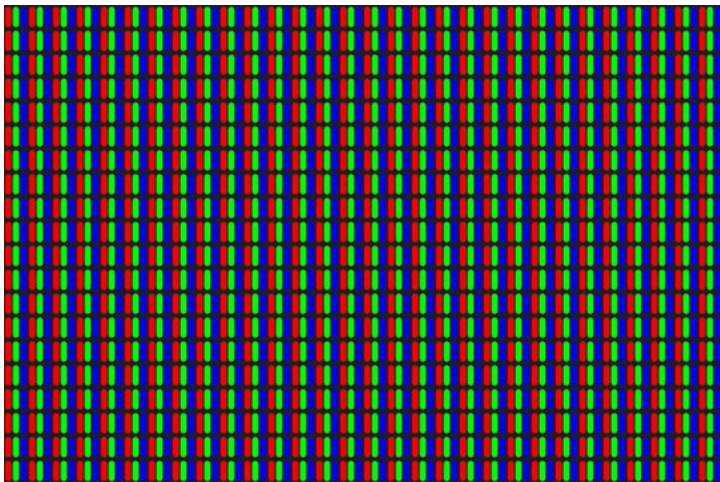
Outline



- การซ้อน Loop
- แนวทางการประยุกต์ใช้การซ้อน Loop
 - การจัดการข้อมูลในรูปแบบตารางหรือเมทริกซ์
 - การจัดการรูปภาพ
- ตัวอย่างโจทย์

แนวทางการประยุกต์ใช้การซ้อน Loop: รูปภาพ

- ภาพดิจิทัลที่เราเห็นกันจริงๆ แล้วประกอบขึ้นมาจากจุดที่เรียงตัวกันเป็นตารางขนาดใหญ่อย่างหนาแน่น
- ข้อมูลในแต่ละช่อง แท้จริงแล้วคือองค์ประกอบของภาพ
- ถ้าเลือกใส่ค่าสีที่เหมาะสมก็จะเห็นเป็นภาพได้



แนวทางการประยุกต์ใช้การซ้อน Loop: รูปภาพ

- ในตอนนี้เราจะยังไม่ทำไปถึงจุดนั้น เพราะการเล่นกับการทำงานรูปภาพจะต้องใช้หลักการของโครงสร้างข้อมูลจำพวกตารางหรือแถวข้อมูลเสียก่อน (อาร์เรย์) ซึ่งจะได้เรียนกันต่อไป
- ตอนนี้เราจะใช้แค่ loop 2 ชั้นเท่าที่ทำได้ไปก่อน

Outline



- การซ้อน Loop
- แนวทางการประยุกต์ใช้การซ้อน Loop
 - การจัดการข้อมูลในรูปแบบตารางหรือเมทริกซ์
 - การจัดการรูปภาพ
- ตัวอย่างโจทย์

ตัวอย่างโจทย์ (1)



โปรแกรมนี้จะพิมพ์เลขอะไรออกมาบ้าง

```
#include <stdio.h>
void main() {
    int sum = 0;
    for(int i = 0; i < 7; ++i) {
        for(int j = 0; j < 3; ++j) {
            sum = sum + 1;
        }
        printf("%d ", sum);
    }
}
```

ตัวอย่างโจทย์ (1)



โปรแกรมนี้จะพิมพ์เลขอะไรออกมาบ้าง

```
#include <stdio.h>
void main() {
    int sum = 0;
    for(int i = 0; i < 7; ++i) {
        for(int j = 0; j < 3; ++j) {
            sum = sum + 1;
        }
        printf("%d ", sum);
    }
}
```

3 6 9 12 15 18 21

ตัวอย่างโจทย์ (2)



โปรแกรมนี้จะพิมพ์เลขอะไรออกมาบ้าง

```
#include <stdio.h>
void main() {
    for(int i = 0; i < 7; ++i) {
        for(int j = 0; j < 3; ++j) {
            printf("%d ", i + j);
        }
    }
}
```

ตัวอย่างโจทย์ (2)



โปรแกรมนี้จะพิมพ์เลขอะไรออกมาบ้าง

```
#include <stdio.h>
void main() {
    for(int i = 0; i < 7; ++i) {
        for(int j = 0; j < 3; ++j) {
            printf("%d ", i + j);
        }
    }
}
```

0 1 2 1 2 3 2 3 4 3 4 5 4 5 6 5 6 7 6 7 8

ตัวอย่างโจทย์ (3)



พิมพ์ตัวเลขขั้นบันได

- เพื่อที่จะเรียนรู้แนวคิดอุปสองชั้น เรามาดูตัวอย่างเพิ่มเติม
- สมมติว่าผู้ใช้ใส่เลข 5 เข้ามาแล้วเราต้องการพิมพ์ว่า

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

- ถ้าผู้ใช้ใส่เลข 10 เข้ามา โปรแกรมก็ต้องไล่ไปจนถึง 10 ถ้าใส่จำนวนเต็มบวก N เข้ามา ก็ต้องไล่ไปเรื่อย ๆ จนถึง N
- แสดงว่าต้องมี N แถว ส่วนจำนวนคอลัมน์ก็ตรงกับหมายเลขแถวที่โปรแกรมกำลังพิมพ์นั่นเอง

ตัวอย่างโจทย์ (3)

พิมพ์ตัวเลขชั้นบันได

- ลองกำหนดเลขแถวตายตัวไว้ที่ `int row = 4`; ก่อน แสดงว่าเราจะพิมพ์เลข 1 2 3 4

```
#include <stdio.h>
void main() {
    int row = 4;
    for(int col = 1; col <= row; ++col) {
        printf("%d ", col);
    }
    printf("\n");
}
```



- จากนั้น เราต้องเอาลูปอีกชั้นมาครอบเพื่อให้มันเปลี่ยนเลขแถวได้อย่างที่ควรจะเป็น

ตัวอย่างโจทย์ (3)



พิมพ์ตัวเลขขั้นบันได

- โจทย์สไลด์การแสดงผลแบบนี้ให้แบ่งคิดทีละส่วน
- สำหรับข้อนี้ เมื่อรับค่า N เข้ามา ผลลัพธ์ก็ควรมี N บรรทัด เราก็คิดว่าแต่ละบรรทัดเราจะพิมพ์อะไรออกมา
- อย่าพยายามคิดรวบเดียวจบ มันยาก

ข้อมูลเข้า → 5
ข้อมูลออก →

1				
1	2			
1	2	3		
1	2	3	4	
1	2	3	4	5

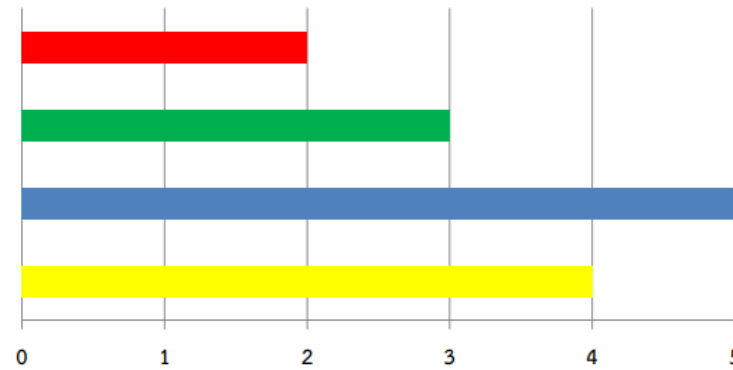
```
#include <stdio.h>
void main() {
    int N;
    scanf("%d", &N);
    for(int row = 1; row <= N; ++row) {
        for(int col = 1; col <= row; ++col) {
            printf("%d ", col);
        }
        printf("\n");
    }
}
```

ตัวอย่างโจทย์ (4)



กราฟแท่งแนวนอน

- กราฟแท่งแนวนอน มีลักษณะเป็นแบบนี้ (จากซีต อ.ภิญโญ แท้ประสาทสิทธิ์)



- เราจะแทนค่าออกมาด้วยจำนวนเครื่องหมาย * จากข้างบนไปเป็นแบบนี้แทน

* *

* * *

* * * * *

* * * *

ตัวอย่างโจทย์ (4)

กราฟแท่งแนวนอน

ข้อมูลเข้า เป็นเลขจำนวนเต็มบวกหรือ 0 จำนวน N ค่าที่บอกความยาวของกราฟแท่งจำนวน N แท่ง จุดสิ้นสุดของข้อมูลคือเลขจำนวนเต็มลบ

ตัวอย่างข้อมูลขาเข้า 2 3 5 4 -1

ผลลัพธ์

กราฟแท่งแนวนอน หนึ่งแท่งต่อหนึ่งบรรทัด แต่ละแท่งประกอบด้วยเครื่องหมาย * มีจำนวนตามตัวเลขแต่ละค่าที่ผู้ใช้ใส่เข้ามา

ตัวอย่างผลลัพธ์

```
* *  
* * *  
* * * * *  
* * * *
```

ตัวอย่างโจทย์ (4)



กราฟแท่งแนวนอน

เรารู้ว่าจากตัวเลขแต่ละตัว เราจะต้องพิมพ์ดอกจันออกมาตามค่าตัวเลขนั้น

- นั่นคือเราต้องวนลูปพิมพ์ดอกจันตามจำนวนรอบที่ถูกระบุด้วยตัวเลขดังกล่าว
- เมื่ออ่านเลขเข้ามา โปรแกรมก็ต้องวนลูปทำนองนี้ เพื่อพิมพ์ดอกจัน
- แต่ถ้าทำแค่นี้ จะรับเลขได้ตัวเดียว แล้วพิมพ์เครื่องหมายดอกจันและจบโปรแกรมเลย

```
#include <stdio.h>
void main() {
    int k;
    scanf("%d", &k);
    for(int i = 0; i < k; ++i) {
        printf("*");
    }
    printf("\n");
}
```

```
5
*****
```

```
2
**
```

```
6
*****
```

```
5 4
*****
```

ตัวอย่างโจทย์ (4)



กราฟแท่งแนวนอน

สิ่งที่ควรทำต่อก็คือ ครอบด้วย loop อีกชั้นหนึ่ง เพื่อให้รับค่าซ้ำ ๆ ได้ ดังนี้

ใช้ while เป็น loop นอกก็ได้ ตรงนี้กำหนดให้วนซ้ำไปเรื่อย ๆ จนกว่าจะเจอเลขที่น้อยกว่า 0 จึงจะ break;

Loop ในยังคงไว้แบบเดิม ดังนั้นตอนนี้ loop ในจะถูกเรียกใช้งานซ้ำ ๆ ได้แล้ว

```
#include <stdio.h>
void main() {
    while(1) {
        int k;
        scanf("%d", &k);
        if(k < 0)
            break;
        for(int j = 0; j < k; ++j) {
            printf("*");
        }
        printf("\n");
    }
}
```

ตัวอย่างโจทย์ (4)

กราฟแท่งแนวนอน

ตอนนี้เราสามารถอินพุตเลขเข้าไปก็ตัวก็ได้ (ตัวสุดท้ายต้องติดลบ)

```
#include <stdio.h>
void main() {
    while(1) {
        int k;
        scanf("%d", &k);
        if(k < 0)
            break;
        for(int j = 0; j < k; ++j) {
            printf("*");
        }
        printf("\n");
    }
}
```

```
9 1 5 7 2 1 4 3 -1
*****
*
*****
*****
**
*
****
***
```

```
3 2 4 0 5 1 -1
***
**
****
*****
*
```

```
7 3 -1
*****
***
```

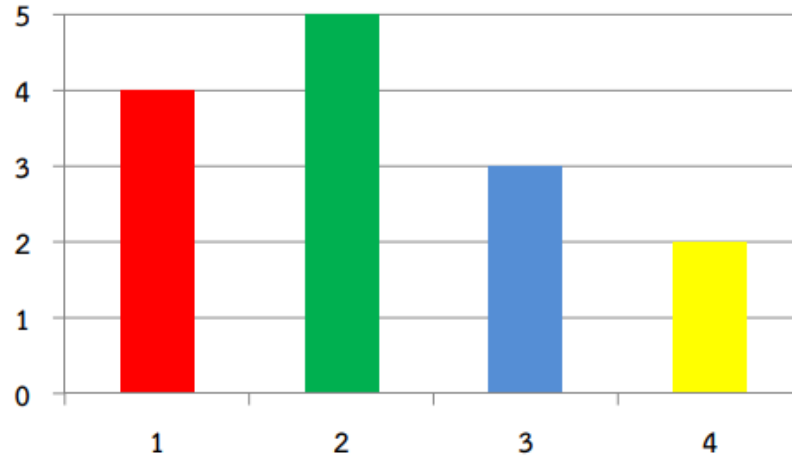
Note: โค้ดนี้เวลากรอกต้องใช้ Space bar เพราะถ้าใช้ enter จะพิมพ์ * สลับกับรับค่าใหม่ทุกครั้ง
คำสั่ง break; อยู่ในชั้นใด จะทำการหยุด loop ในชั้นนั้น

ตัวอย่างโจทย์ (5)



กราฟแท่งแนวนั่ง

เปลี่ยนจากตัวอย่างที่ (4) ให้เป็นกราฟแนวนั่ง สามารถทำได้หรือเปล่า



ตัวอย่างโจทย์ (5)



กราฟแท่งแนวตั้ง

ตอบ ตอนนี้ยังทำไม่ได้ (ถ้ารู้เรื่องอาเรย์แล้วจะทำได้)

เพราะกว่าจะรู้ว่าควรตั้งฐานกราฟไว้ที่บรรทัดใด เราก็ต้องอ่านข้อมูลให้ครบก่อน เพื่อที่จะหากราฟแท่งที่สูงที่สุด

- แต่พอรู้ข้อมูลตรงนี้แล้ว ก็จะมีปัญหาว่าค่าตัวเลขที่อ่านมาไม่ได้ถูกเก็บไว้
- จะสร้างตัวแปรมาเก็บไว้ก็คงต้องสร้างมาเป็นร้อย ถ้าแท่งมันมีเป็นร้อย
- ปัญหาทำนองนี้เราควรใช้อาเรย์เข้าช่วยเพื่อรวมค่าตัวแปรต่าง ๆ ลงเป็นชุดเดียวกัน และใช้ไคด์เดียวกันให้ได้

ตัวอย่างโจทย์ (5)



กราฟแท่งแนวตั้ง

ตอบ ตอนนี่ยังทำไม่ได้ (ถ้ารู้เรื่องอาเรย์แล้วจะทำได้)

แต่สมมติว่าเราจะฝืน! เจอข้อสอบทำเคสยักษ์ๆไม่ได้ แต่ขอซักเคสเล็กๆสองสามเคสก็ยังดี
ลองเคสที่มีแค่ 3 แท่งก่อน ถ้าแบบนี้น่าจะพอทำได้ เพราะไม่ต้องประกาศตัวแปรเป็นร้อยตัว

ตัวอย่างโจทย์ (5)

กราฟแท่งแนวตั้ง 3 แท่ง

- กำหนดให้กราฟแท่งแนวตั้งมีทั้งหมด 3 แท่ง มีความสูง x , y , และ z ค่าเหล่านี้เป็นจำนวนเต็มบวกหรือศูนย์ (ศูนย์คือเป็นแท่งเปล่า)
- กราฟทั้งสามแท่งต้องมีฐานจากบรรทัดเดียวกัน และบรรทัดแรกจะต้องมีดอกจันของกราฟแท่งที่สูงที่สุดอยู่ด้วย (กล่าวคือห้ามมีบรรทัดเปล่า)

แนวคิด

1. แบบนี้ก็แสดงว่าเราจะต้องหาให้ได้ก่อนว่าความสูงของกราฟที่สูงที่สุดคือเท่าใด
2. ต้องคำนวณให้ได้ว่ากราฟแต่ละแท่งจะมีช่องว่างกี่บรรทัดจนกว่าจะมีดอกจันเป็นอันแรก
3. จำนวนบรรทัดเปล่าที่ว่าเป็นสิ่งที่สัมพันธ์กับกราฟแท่งที่สูงที่สุด

ตัวอย่างโจทย์ (5)



กราฟแท่งแนวนั่ง **3 แท่ง**

พิจารณากราฟตัวอย่าง เมื่อ $x=2$, $y=5$ และ $z=3$

	คอลัมน์ 1	คอลัมน์ 2	คอลัมน์ 3
แถว 1		*	
แถว 2		*	
แถว 3		*	*
แถว 4	*	*	*
แถว 5	*	*	*

ตัวอย่างโจทย์ (5)



กราฟแท่งแนวตั้ง **3 แท่ง** : 1. หาความสูงแท่งที่สูงที่สุด

```
int x, y, z;  
scanf("%d %d %d", &x, &y, &z);  
int max = INT_MIN;  
if(x > max)  
    max = x;  
if(y > max)  
    max = y;  
if(z > max)  
    max = z;
```

ตัวอย่างโจทย์ (5)

กราฟแท่งแนวตั้ง **3 แท่ง** : 2. พิมพ์แท่งกราฟ

คำถามชวนคิด เวลาจะวาดกราฟแต่ละแท่ง เราต้องไล่เรียงทีละบรรทัด แล้วเราจะรู้ได้อย่างไรว่า ในแถวที่กำลังดำเนินการอยู่ กราฟแท่งนี้ต้องถูกเขียนด้วยช่องว่างหรือดอกจัน?

พิจารณากราฟตัวอย่าง เมื่อ $x=2$, $y=5$ และ $z=3$
สังเกตเห็นหรือไม่ว่า ความแตกต่างระหว่างความสูงของแท่งที่สูงสุดกับแท่งที่เราพิจารณาก็คือจำนวนช่องว่างของแท่งกราฟนั้น ๆ (เช่น แท่งแรกมีช่องว่าง 3 แถว เพราะ $5 - 2 = 3$)

```

      *
      *
    *  *
  *  *  *
  *  *  *
  
```

	คอลัมน์ 1	คอลัมน์ 2	คอลัมน์ 3
แถว 1		*	
แถว 2		*	
แถว 3		*	*
แถว 4	*	*	*
แถว 5	*	*	*

ตัวอย่างโจทย์ (5)



กราฟแท่งแนวตั้ง 3 แท่ง : 2. พิมพ์แท่งกราฟ

- เราต้องไล่ไปที่ละแถว เพราะมันเป็นข้อจำกัดของการพิมพ์ผลลัพธ์
- จากการวิเคราะห์ในหน้าที่แล้ว ถ้าหากเอาค่าสูงสุดลบด้วยความสูงของแท่งที่สนใจ จะได้จำนวนแถวที่เป็นช่องว่าง
- ถ้าหมายเลขแถวเกินจุดนี้ไป จะต้องพิมพ์ *

	คอลัมน์ 1	คอลัมน์ 2	คอลัมน์ 3
แถว 1		*	
แถว 2		*	
แถว 3		*	*
แถว 4	*	*	*
แถว 5	*	*	*

ตัวอย่างโจทย์ (5)

กราฟแท่งแนวตั้ง 3 แท่ง : 2. พิมพ์แท่งกราฟ

```
for(int row = 1; row <= max; ++row) {  
    if(row > max - x) printf("*");  
    else printf(" ");  
    if(row > max - y) printf("*");  
    else printf(" ");  
    if(row > max - z) printf("*");  
    else printf(" ");  
    printf("\n");  
}
```

ตัวอย่างโจทย์ (5)

กราฟแท่งแนวตั้ง 3 แท่ง : พอรวมทั้งโปรแกรมจะได้ดังนี้

ในกรณีของตัวอย่างที่ถูกแปลงให้ง่ายลงนี้เป็นรูปขั้นเดียว

- เพราะเราเขียนโค้ดซ้ำ ๆ ตรงค่าตัวแปร x, y, และ z
- ถ้าเรามีร้อยแท่งก็คงจะต้องมี x1, x2, x3, ..., x100
- แต่การใช้ arrays เราจะผนวกตัวแปรเข้าภายใต้ชื่อเดียวกันได้ และทำให้เราสามารถที่จะเปลี่ยนโค้ดที่ดูซ้ำ ๆ ไปเป็นรูปแทน

เดียวเราจะมาดูอีกครั้งตอนเรียนเรื่อง arrays แล้ว

แล้วมาทำภาคสมบูรณ์ไปด้วยกัน

```
#include <stdio.h>
#include <limits.h>
void main() {
    int x, y, z;
    scanf("%d %d %d", &x, &y, &z);
    int max = INT_MIN;
    if(x > max)
        max = x;
    if(y > max)
        max = y;
    if(z > max)
        max = z;

    for(int row = 1; row <= max; ++row) {
        if(row > max - x) printf("*");
        else printf(" ");
        if(row > max - y) printf("*");
        else printf(" ");
        if(row > max - z) printf("*");
        else printf(" ");
        printf("\n");
    }
}
```

```
2 5 4
*
**
**
***
***
***
```

ตัวอย่างโจทย์ (6)



พิมพ์กรอบสี่เหลี่ยม

สมมติว่าผู้ใช้ใส่เลข 7 เข้ามาแล้วเราอยากได้กรอบตามแบบข้างล่างนี้

```
* * * * *
*           *
*           *
*           *
*           *
*           *
* * * * *
```

- ขอให้สังเกตให้ดีว่ามันมีบางช่วงของงานที่เหมือนกัน และบางช่วงที่ต่างกัน
- รูปของเราอาจจะต้องแบ่งเป็นหลาย ๆ ส่วน เราจะใช้รูปที่เหมือนกันไปเสียทุกอย่างก็คงจะไม่ใช่วิธีเลือกที่ดี
- หมายเหตุ ดอกจันในบรรทัดแรกและสุดท้ายถูกคั่นด้วยช่องว่างหนึ่งช่อง

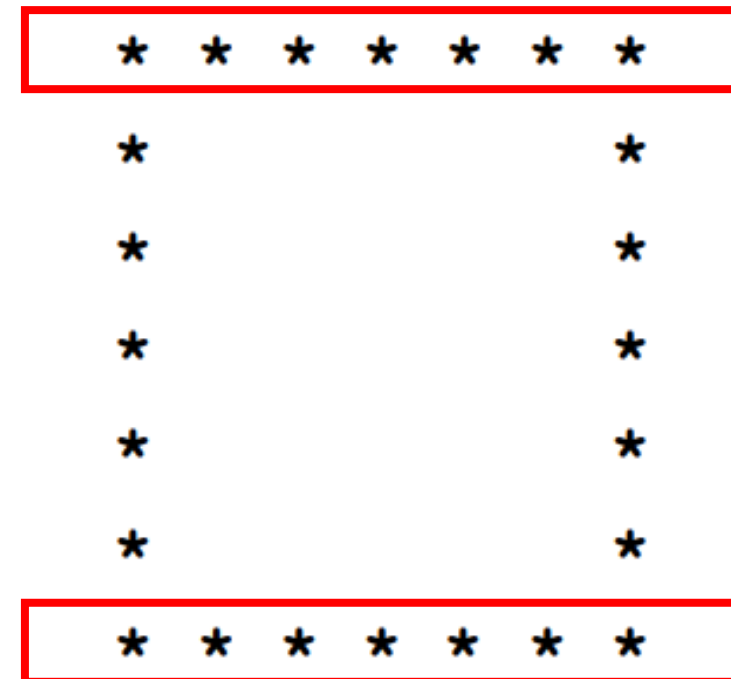
ตัวอย่างโจทย์ (6)



พิมพ์กรอบสี่เหลี่ยม : แยกงานเป็นส่วน ๆ

โจทย์ที่มีความซับซ้อนขึ้น การแยกงานออกเป็นส่วน ๆ จะทำให้มองโจทย์ง่ายขึ้น

- บรรทัดแรกกับบรรทัดสุดท้ายเหมือนกัน



ตัวอย่างโจทย์ (6)

พิมพ์กรอบสี่เหลี่ยม : แยกงานเป็นส่วน ๆ

โจทย์ที่มีความซับซ้อนขึ้น การแยกงานออกเป็นส่วน ๆ จะทำให้มองโจทย์ง่ายขึ้น

- บรรทัดแรกกับบรรทัดสุดท้ายเหมือนกัน
- ตรงกลางจะเป็นอีกกลุ่มหนึ่งที่ทำงานไม่เหมือนบรรทัดแรกและบรรทัดสุดท้าย (แต่จะเหมือนบรรทัดตรงกลางด้วยกันทุกประการ)

* เริ่มเห็นอะไรซ้ำ ๆ บ้างหรือยัง



ตัวอย่างโจทย์ (6)

พิมพ์กรอบสี่เหลี่ยม : แยกงานเป็นส่วน ๆ

ดังนั้นงานจะถูกออกแบบมาเป็นสามส่วน

1. พิมพ์บรรทัดแรก
2. พิมพ์บรรทัดช่วงกลาง
3. พิมพ์บรรทัดสุดท้าย (ใช้โค้ดเดียวกับของบรรทัดแรกได้)



ตัวอย่างโจทย์ (6)

พิมพ์กรอบสี่เหลี่ยม : 1. รับข้อมูลความยาวกรอบ และพิมพ์บรรทัดแรก

- โค้ดนี้พิมพ์ดอกจันขึ้นด้วยช่องว่างและจบท้ายด้วยการขึ้นบรรทัดใหม่
- เราพิมพ์จำนวนดอกจันออกมาเป็นจำนวน N ค่า ในแถวแรก

```
int N;  
scanf("%d", &N);  
for(int col = 1; col <= N; ++col) {  
    printf("* ");  
}  
printf("\n");
```

```
5  
* * * * *
```

ตัวอย่างโจทย์ (6)



พิมพ์กรอบสี่เหลี่ยม : 2. พิมพ์แถวตรงกลาง

- จากทั้งหมด N แถว เราจะพิมพ์แถวที่ 2 ถึง $N - 1$ ในลักษณะที่มีเฉพาะดอกจันตรงขอบซ้ายขวา
- ดังนั้นตำแหน่งคอลัมน์แรกและคอลัมน์สุดท้ายในแถวจึงเป็นจุด นอกนั้นเป็นเว้นวรรค
- ไม่ต้องใช้ลูป (วิธีที่จะใช้ loop ก็มีเหมือนกัน แล้วแต่สไตล์)
- ส่วนตรงกลางเป็นของที่เหมือนกัน ๆ ใช้การวนซ้ำได้

ตัวอย่างโจทย์ (6)



พิมพ์กรอบสี่เหลี่ยม : 2. พิมพ์แถวตรงกลาง

```
for(int row = 2; row <= N - 1; ++row) {  
    printf("* ");  
    for(int col = 2; col < N; ++col) {  
        printf(" ");  
    }  
    printf("*\n");  
}
```

ผลจากโค้ดส่วนแรก

```
5  
* * * * *  
*           *  
*           *  
*           *
```

ผลจากโค้ดส่วนนี้

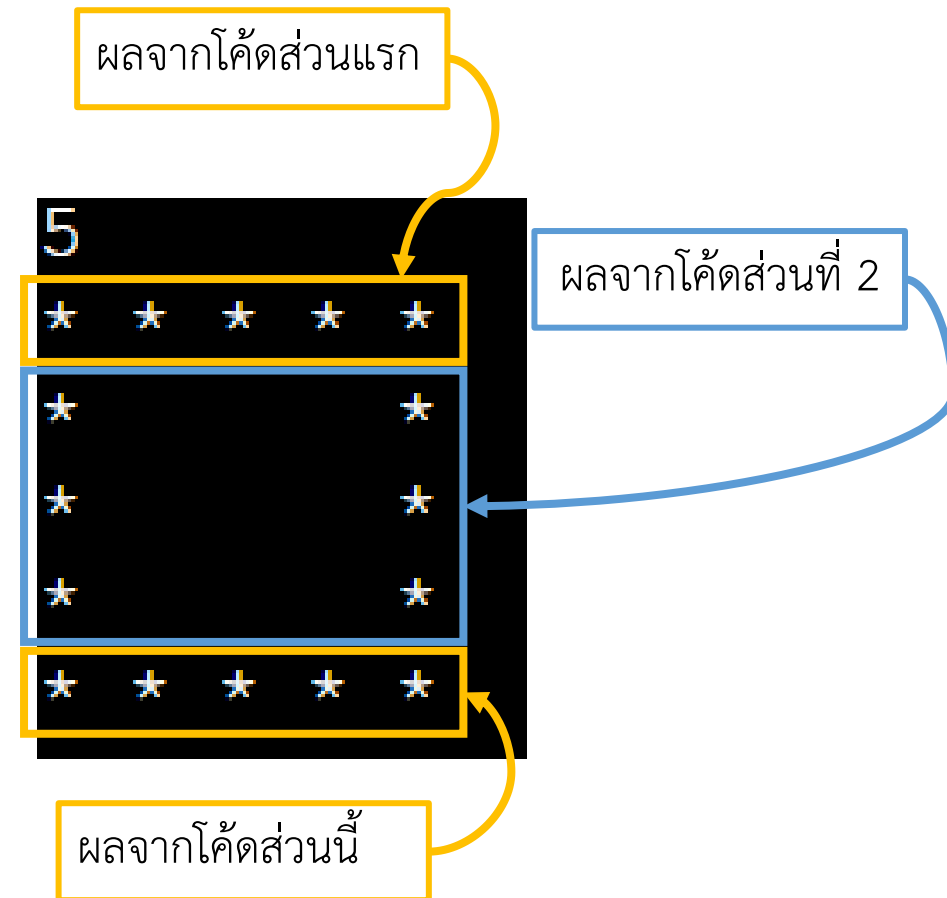
ตัวอย่างโจทย์ (6)



พิมพ์กรอบสี่เหลี่ยม : 3. พิมพ์แถวสุดท้าย (เอาโค้ดแถวแรกมาแปะ)

- แต่ไม่ต้องมีการขึ้นบรรทัดใหม่เหมือนแถวแรก

```
int N;  
scanf("%d", &N);  
for(int col = 1; col <= N; ++col) {  
    printf("* ");  
}
```



ตัวอย่างโจทย์ (6)



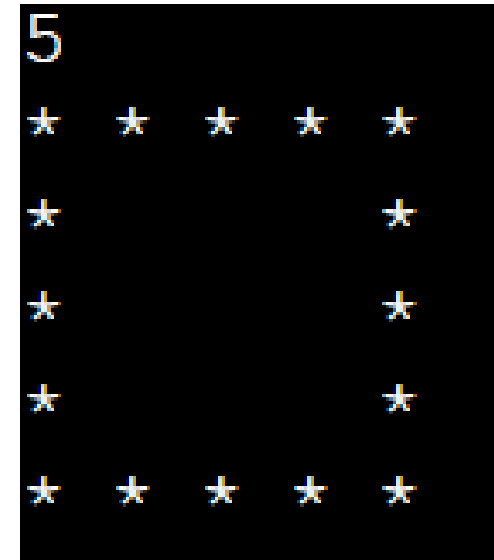
พิมพ์กรอบสี่เหลี่ยม : 1.-3. ฟิวชั่น

```
#include <stdio.h>
void main() {
    int N;
    scanf("%d", &N);
    for(int col = 1; col <= N; ++col) {
        printf("* ");
    }
    printf("\n");

    for(int row = 2; row <= N - 1; ++row) {
        printf("* ");
        for(int col = 2; col < N; ++col) {
            printf(" ");
        }
        printf("*\n");
    }

    for(int col = 1; col <= N; ++col) {
        printf("* ");
    }

}
```



สรุป



- งานที่ใช้ loop สองชั้นมีลำดับการคิดคล้ายกับ loop ชั้นเดียว (พยายามมอง loop ให้เป็นคำสั่งธรรมดาตัวหนึ่ง)
- เมื่อรันไปจนถึง loop ด้านใน ก็จะมีการวนซ้ำ loop ตัวนั้นให้เรียบร้อยหมดก่อนจึงจะหลุดออกจาก loop และทำคำสั่งต่อไป และเมื่อสำเร็จทุกคำสั่งใน loop นอก ก็จะสามารถวนขึ้นไปตรวจสอบเงื่อนไขต่อที่ลูปด้านนอกได้
- เวลาที่จินตนาการไม่ออกให้มองว่าลูปด้านในเป็นเหมือนงานบรรทัดหนึ่งที่ทำเสร็จแล้วก็ข้ามไปทำบรรทัดถัดไป ด้วยเหตุนี้เราจึงอาจจะลองเริ่มคิดจากเนื้อหาของลูปด้านในให้เสร็จก่อน แล้วค่อยเอาลูปด้านนอกมาครอบอีกชั้น
- ถ้างานมันมีลักษณะแบ่งเป็นหลาย ๆ แบบ ลูปของเราอาจจะมีหลาย ๆ ชุดก็ได้ หรือถ้าจะมีชุดเดียวใหญ่ ๆ ก็อาจจะต้องพึ่งพาการใช้ if จำนวนมากเพื่อแยกประเภทงาน (ให้เลือกทางที่ใช้)