



LOOP

*(while loop, for loop,
do while, nested loop)*

សំបកព័ត៌មានទីរ

while loop

for loop

do while

nested loop

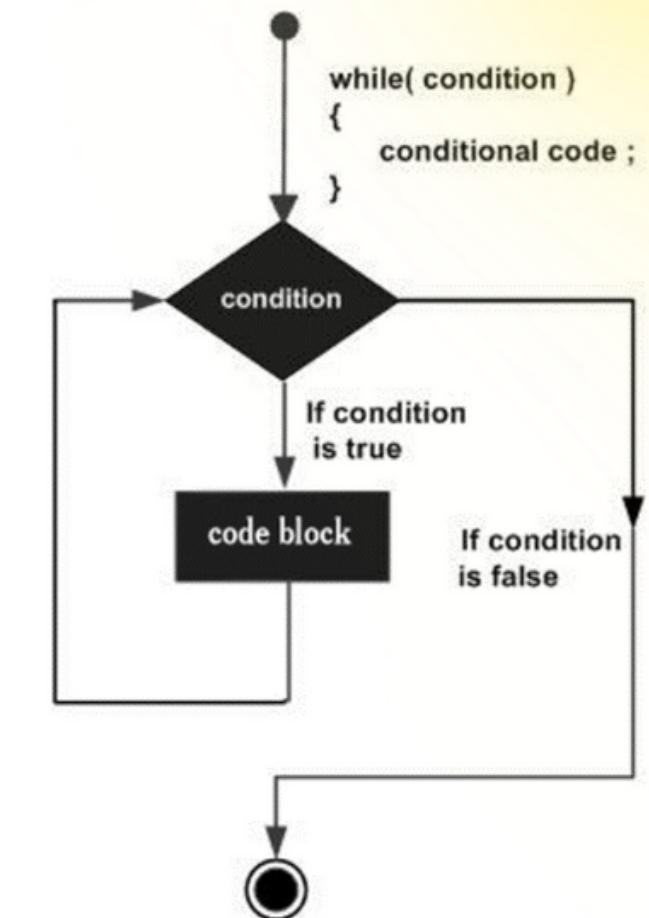
Loop control statement

WHILE LOOP

ໃচ້ໃນກຳງານໜີ້ ແລະ ຮັບຄຳສັ່ງເປົາໝາຍຕຣາບເທົ່າທີ່
ເປັນຄວາມຈົງ

syntaxຂອງ while loop

```
while(condition) {  
    statement(s);  
}
```



ຕັ້ງອອຍ່ານ WHILE LOOP

ຄີດວ່າຜລພຽດໄດ້ຈະໄດ້ ?

```
#include <stdio.h>

int main () {

    /* local variable definition */
    int a = 10;

    /* while loop execution */
    while( a < 20 ) {
        printf("value of a: %d\n", a);
        a++;
    }

    return 0;
}
```

ପାର୍ଶ୍ଵ

```
#include <stdio.h>

int main () {

    /* local variable definition */
    int a = 10;

    /* while loop execution */
    while( a < 20 ) {
        printf("value of a: %d\n", a);
        a++;
    }

    return 0;
}
```

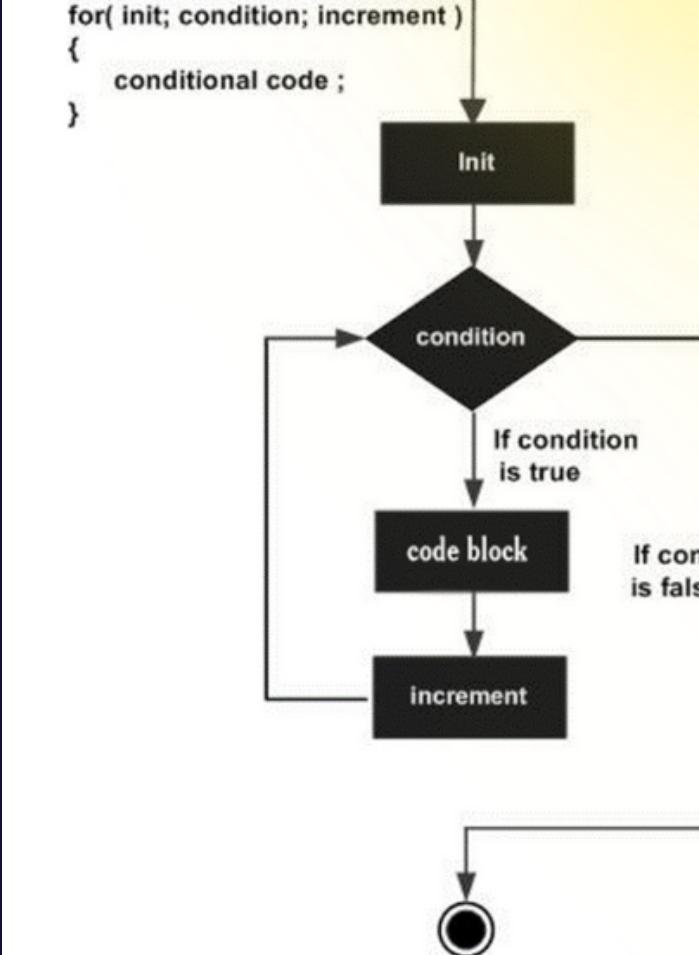
```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

FOR LOOP

ໃຊ້ໃນກຳງານຫຼັງ ແລະ ກຳນົດຈຳນວນຮອບເຈົ້າຈົງໄດ້

syntaxຂອງ for loop

```
for ( init; condition; increment ) {  
    statement(s);  
}
```



ตัวอย่าง ***FOR LOOP***

คิดว่าผลลัพธ์ได้อะไร ?

```
#include <stdio.h>

int main () {

    int a;

    /* for loop execution */
    for( a = 10; a < 20; a = a + 1 ){
        printf("value of a: %d\n", a);
    }

    return 0;
}
```

ପାର୍ଯ୍ୟ

```
#include <stdio.h>

int main () {

    int a;

    /* for loop execution */
    for( a = 10; a < 20; a = a + 1 ){
        printf("value of a: %d\n", a);
    }

    return 0;
}
```

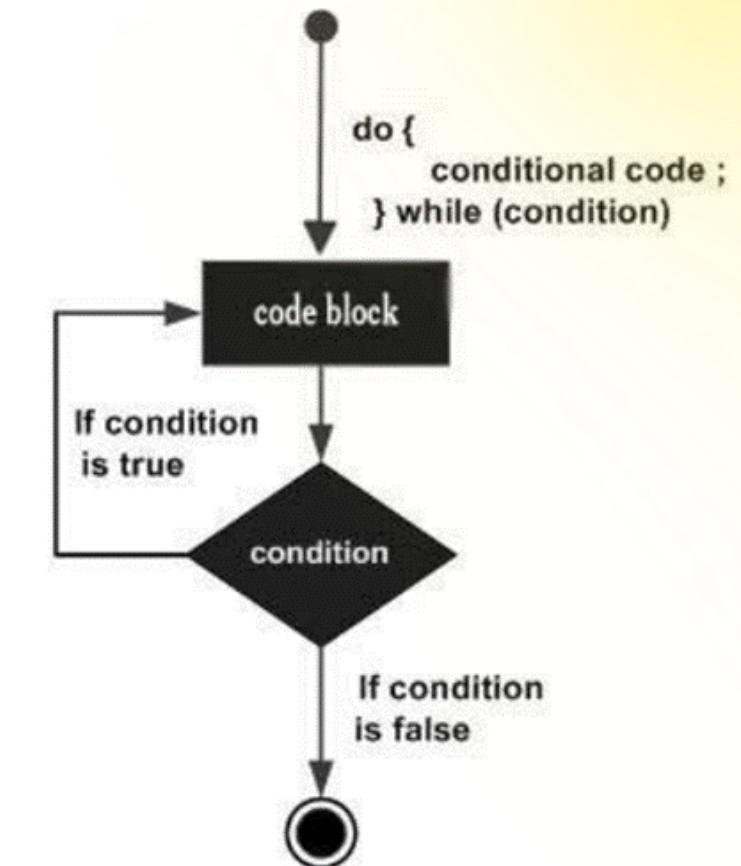
```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

DO WHILE

การเริ่มทำก่อนแล้วค่อยมาตรวจสอบเงื่อนไข หากยังเป็น 'จริง' จึงจะทำซ้ำ

syntaxของ do while

```
do {  
    statement(s);  
} while( condition );
```



ຕັ້ງອຍ່ານ *DO WHILE*

ຄິດວ່າຜລລພຣໄດ້ຈະໄດ້ ?

```
#include <stdio.h>

int main () {

    /* local variable definition */
    int a = 10;

    /* do loop execution */
    do {
        printf("value of a: %d\n", a);
        a = a + 1;
    }while( a < 20 );

    return 0;
}
```

ପାର୍ଶ୍ଵ

```
#include <stdio.h>

int main () {

    /* local variable definition */
    int a = 10;

    /* do loop execution */
    do {
        printf("value of a: %d\n", a);
        a = a + 1;
    }while( a < 20 );

    return 0;
}
```

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

NESTED LOOP

ใช้สำหรับการสร้างหนึ่งลูปภายในอีกวงหนึ่ง

syntax ของ nested loop

```
for ( init; condition; increment ) {  
  
    for ( init; condition; increment ) {  
        statement(s);  
    }  
    statement(s);  
}
```

```
while(condition) {  
  
    while(condition) {  
        statement(s);  
    }  
    statement(s);  
}
```

```
do {  
    statement(s);  
  
    do {  
        statement(s);  
    }while( condition );  
}  
while( condition );
```

ຕັ້ງອຍ່ານ *NESTED LOOP*

ຄິດວ່າຜລລັພນໄດ້ຈະໄດ້ ?

```
#include <stdio.h>

int main () {

    /* local variable definition */
    int i, j;

    for(i = 2; i<100; i++) {

        for(j = 2; j <= (i/j); j++)
            if(!(i%j)) break; // if factor found, not prime
        if(j > (i/j)) printf("%d is prime\n", i);
    }

    return 0;
}
```



```
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
29 is prime
31 is prime
37 is prime
41 is prime
43 is prime
47 is prime
53 is prime
59 is prime
61 is prime
67 is prime
71 is prime
73 is prime
79 is prime
83 is prime
89 is prime
97 is prime
```

LOOP CONTROL STATEMENT

เป็นชุดคำสั่งสำหรับดำเนินในลูป เช่น การสั่งให้หยุด กำหนด ไปที่ หรือ กำสิ่งที่ส่งผลกระทบต่อวงของกระบวนการช้านัน

แบ่งเป็น 3 อาย่าง

Break

Continue

Goto

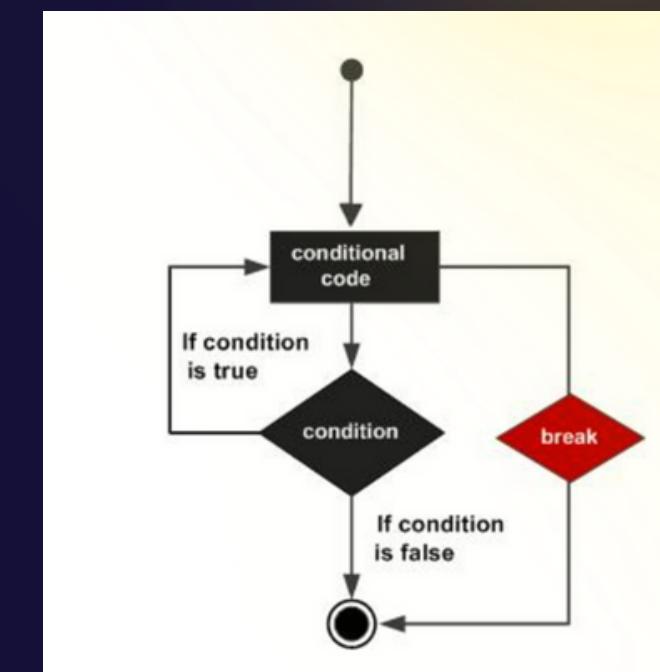
BREAK

`break` คำสั่งในการเขียนโปรแกรม C มีสังत่อไปนี้

- เมื่อพิมพ์คำสั่ง `break` กายในลูป การวนซ้ำจะสิ้นสุดลง กับกี และการควบคุมโปรแกรมจะกลับมาทำงานต่อในคำ สั่งถัดไปหลังจาก วนซ้ำ
- สามารถใช้เพื่อยุติการวนและปัณหาในคำสั่ง `switch` (ครอบคลุมในบทต่อไป)

syntaxของ `break`

```
break;
```



ຕັ້ງອຍ່ານ **BREAK**

ຄີດວ່າຜລລັພນໄດ້ຈະໄດ້ ?

```
#include <stdio.h>

int main () {

    /* local variable definition */
    int a = 10;

    /* while loop execution */
    while( a < 20 ) {

        printf("value of a: %d\n", a);
        a++;

        if( a > 15) {
            /* terminate the loop using break statement */
            break;
        }
    }

    return 0;
}
```



```
value of a: 10  
value of a: 11  
value of a: 12  
value of a: 13  
value of a: 14  
value of a: 15
```

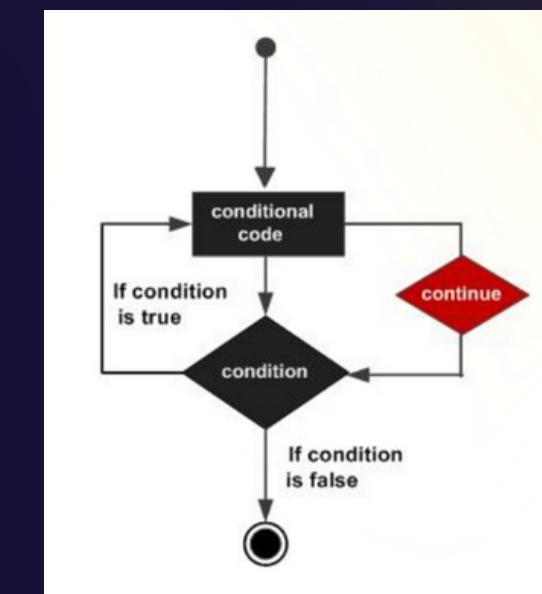
CONTINUE

continue คำสั่งในการเขียนโปรแกรม C ทำงานคล้าย break คำสั่ง แทนที่จะบังคับให้สิ้นเชิง แต่เป็นการบังคับให้วนซ้ำรอบตัวไปเรื่อยๆ โดยข้ามโค้ดใดๆ ที่อยู่ระหว่างนั้น

สำหรับ for loop , คำสั่ง continue เป็นสาเหตุที่ทำให้การทดสอบและการเพิ่มเงื่อนไขบางส่วนของวงล้อทำงานได้ไม่ถูกต้อง สำหรับ while และ do...while loops คำสั่ง continue ทำให้เกิดการควบคุมโปรแกรมที่จะส่งผ่านไปยังการทดสอบเงื่อนไข

syntax ของ continue

```
continue;
```



ຕັ້ງອຍ່ານ *CONTINUE*

ຄົດວ່າຜລລັພຣໄດ້ຈະໄດ້ ?

```
#include <stdio.h>

int main () {

    /* local variable definition */
    int a = 10;

    /* do loop execution */
    do {

        if( a == 15) {
            /* skip the iteration */
            a = a + 1;
            continue;
        }

        printf("value of a: %d\n", a);
        a++;

    } while( a < 20 );

    return 0;
}
```



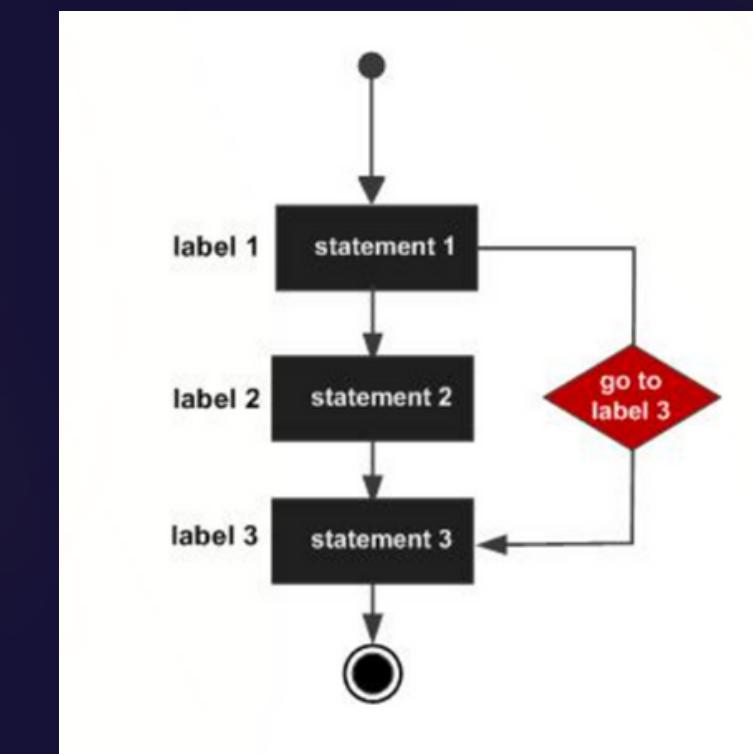
```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

GOTO

goto คำสั่งในการเขียนโปรแกรม C ให้กระโดดไปมีเงื่อนไขจาก 'ไปที่' จะเป็นคำสั่งที่มีข้อความในฟังก์ชันเดียวกัน (ไม่นิยมใช้เท่าไหร่)

syntax ของ goto

```
goto label;  
.  
. .  
label: statement;
```



ຕັ້ງອຍ່ານ GOTO

ຄີດວ່າຜລລັພນໄດ້ຈະໄດ້ ?

```
#include <stdio.h>

int main () {

    /* local variable definition */
    int a = 10;

    /* do loop execution */
    LOOP:do {

        if( a == 15) {
            /* skip the iteration */
            a = a + 1;
            goto LOOP;
        }

        printf("value of a: %d\n", a);
        a++;

    }while( a < 20 );

    return 0;
}
```



```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

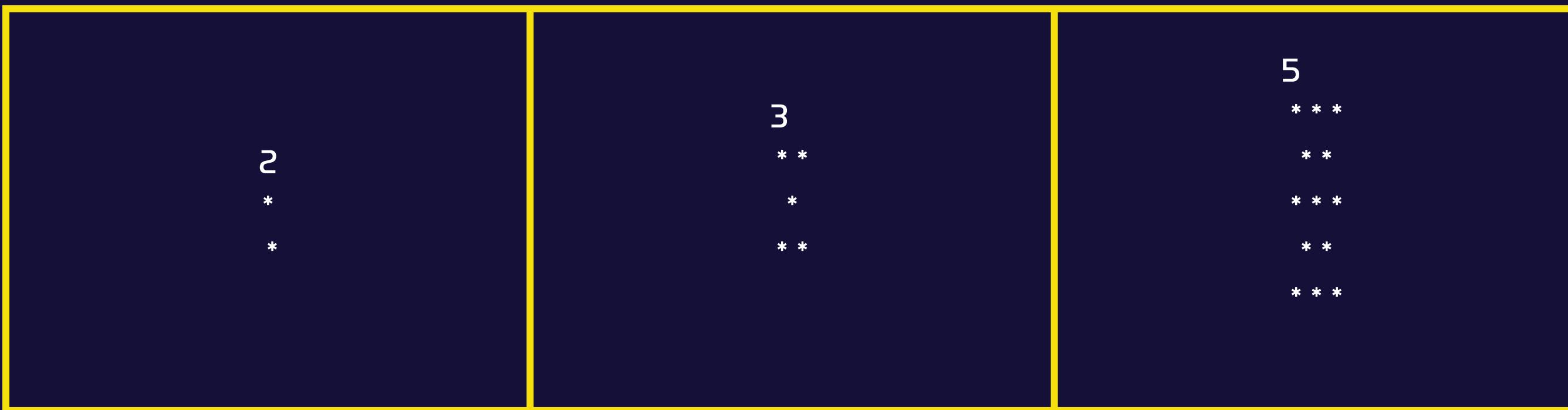
LET'S TRY

จะเขียนโปรแกรมเพื่อรับข้อมูลชนิดตัวเลข และตรวจสอบว่า
ตัวเลขที่ผู้ใช้ใส่เข้ามานั้นเป็นจำนวนเฉพาะหรือไม่ เมื่อแสดง
ผลลัพธ์แล้วให้กลับไปรอรับข้อมูลใหม่จนอย่างนี้ไปเรื่อยๆ
จนกระทั่งผู้ใช้ใส่ค่า -99 จึงจบการทำงาน

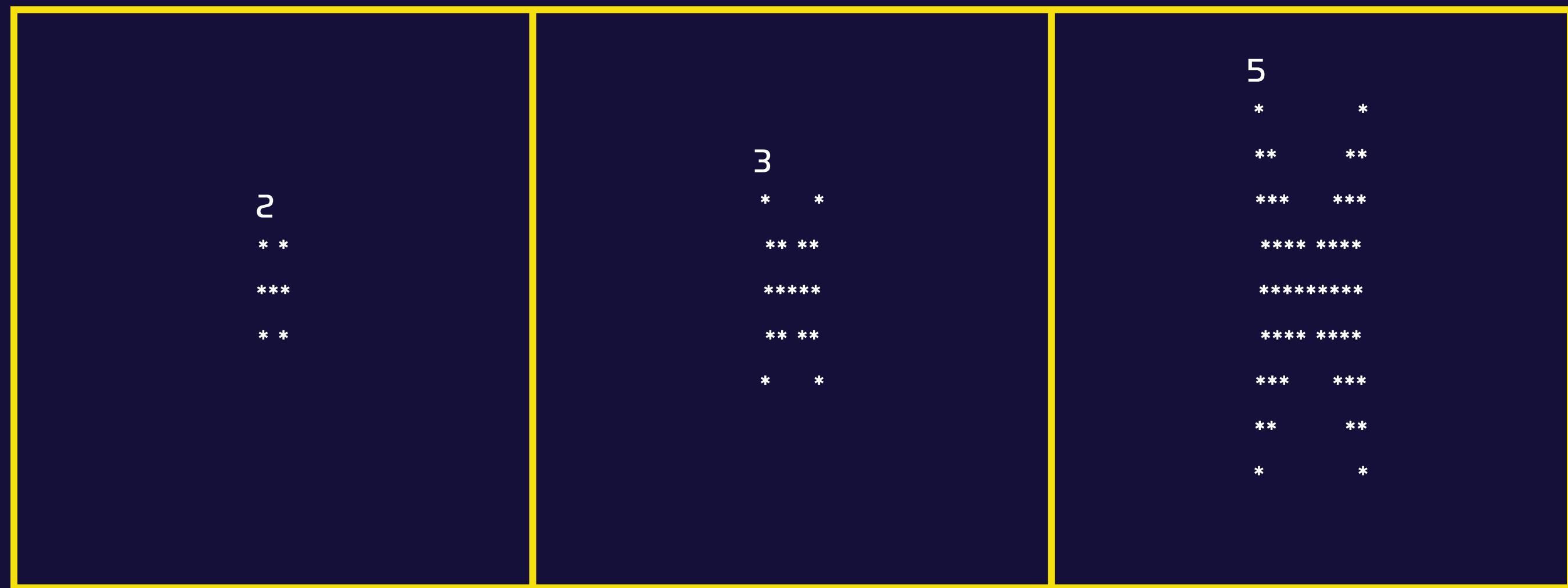
ຈົງເຂັ້ມໂປຣແກຣມຮັບຕົວເລຂ 1 ຕົວແລ້ວໃຫ້ຄອມພິວເຕອົກວາດຽບ
ເຄຣືອງໜາຍ * ເປັນຈຳນວນເທົ່າກັບຕົວເລຂທີ່ຮັບເຂົາມາ ດັ່ງຕົວຍ່າງ
(Level 1)

2 **	5 *****	10 *****
---------	------------	-------------

จงเขียนโปรแกรมรับตัวเลข 1 ตัวแล้วให้คอมพิวเตอร์วาดรูปเครื่องหมาย * เป็นสี่เหลี่ยมที่มีด้านเท่ากับตัวเลขที่รับเข้ามาโดยมีพื้นที่เป็นตารางขนาด กดตัวอักษร (Level 2)



จงเขียนโปรแกรมรับตัวเลข 1 ตัวแล้วให้คอมพิวเตอร์วาดรูปเครื่องหมาย *
เป็นรูปผีเสื้อที่มีขนาดของปีกแต่ละข้างเท่ากับตัวเลขที่รับเข้ามา ดังตัวอย่าง
(Level 3)



ຈິງແສດງ
ຜລຕາມຕັວອຢ່າງ
(Level 4)

input	output
4	-*- *-* *-* -*-
5	--*-- -*-* *___* -*-* --*--
5	--*-- -*-* *___* *___* -*-* --*--

THANK YOU



borntoDev