Chapter 5 – Program Looping

•
••
•••
••••
•••••
••••••
•••••••
••••••••

- สังเกตว่าในแต่ละแถวของพิรามิด จำนวนจุดจะเพิ่มขึ้นหนึ่งจุด
- หากเราต้องการเขียนโปรแกรมนับจำนวนจุดทั้งหมดในพิรามิดนี้ซึ่งมีแปดแถว เราสามารถเขียนได้ดังนี้

Program 5.1 Calculating the Eighth Triangular Number

```
// Program to calculate the eighth triangular number
#include <stdio.h>
int main (void)
{
    int triangularNumber;
    triangularNumber = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8;
    printf("The eighth triangular number is %i\n",
triangularNumber);
    return 0;
}
```

- แต่ถ้าหากจำนวนแถวเพิ่มขึ้นมาก เรามีวิธีที่ง่ายกว่าในการเขียนโปรแกรมเพื่อนับจำนวนจุดหรือไม่

- เราสามารถสั่งคอมพิวเตอร์ให้ทำคำสั่งเดิมซ้ำๆ ได้ เรียกว่าการลูป ทำให้การเขียนโปรแกรมสะดวกขึ้น
- ในภาษา C มีสามคำสั่งที่เกี่ยวข้องกับการลูป คือ for, while, และ do

5.1 คำสั่ง For

Program 5.2 Calculating the 200th Triangular Number

```
/* Program to calculate the 200th triangular number
Introduction of the for statement */
#include <stdio.h>
int main (void)
{
    int n, triangularNumber;
    triangularNumber = 0;
    for(n = 1; n <= 200; n = n + 1)
        triangularNumber = triangularNumber + n;
    printf("The 200th triangular number is %i\n",
triangularNumber);
    return 0;
}
```

- รูปแบบการใช้งานคำสั่ง for คือ

```
for(init_expression; loop_condition; loop_expression)
{
    statement_1;
    statement_2;
    ...
}
```

- init_expression ใช้เซ็ทค่าเริ่มต้นก่อนที่จะเริ่มทำคำสั่งในลูป ในตัวอย่าง n จะถูกกำหนดค่าเริ่มต้นเป็น 1

- loop_condition คือเงื่อนไขที่จะให้ลูปทำงานไปเรื่อยๆ หากผลลัพธ์ของเงื่อนไขเป็น TRUE และหยุดทำงานเมื่อเงื่อนไขเป็น FALSE โดยใน loop_condition นี้จะมีการใช้ Relational operators เข้ามาช่วย ในตัวอย่างนี้ลูปจะทำงานไปเรื่อยๆจนกว่าค่าใน n จะมากกว่า 200
- loop_expression จะถูกกระทำทุกครั้งหลังจากทำคำสั่งในลูปในตัวอย่างนี้ n จะถูกเพิ่มค่าขึ้นหนึ่งค่าหลังจากค่าของมันถูกบวกให้กับ triangularNumber แล้ว
- เราสามารถสรุปการทำงานของลูป for ในตัวอย่างได้ดังนี้
    1. กำหนดให้ n = 1
    2. ทดสอบเงื่อนไขที่จะให้ลูปทำงาน คือ n น้อยกว่าหรือเท่ากับ 200 หรือไม่
    3. ทำคำสั่งที่อยู่ในลูป คือ _____
    4. เพิ่มค่าใน n ขึ้นหนึ่งค่า
    5. กลับไปที่ข้อสอง

Relational Operators

Table 5.1 Relational Operators

Operator	Meaning	Example
==	เท่ากับ	count == 10
!=	ไม่เท่ากับ	flag != 1
<	น้อยกว่า	a < b
<=	น้อยกว่าหรือเท่ากับ	low <= high
>	มากกว่า	pointer > end_of_list
>=	มากกว่าหรือเท่ากับ	j >= 0

- Relational operators เหล่านี้มีความสำคัญน้อยกว่า Operator การคำนวณทางคณิตศาสตร์ ยกตัวอย่างเช่น a < b + c มีความหมายเดียวกับ _____ และเงื่อนไขจะเป็นจริงเมื่อ _____
- สำหรับ operator '==' นั้น มีการใช้งานแตกต่างจาก operator '=' อย่างเช่น 'a==2' จะใช้ทดสอบว่าค่าใน a เท่ากับสองหรือไม่ และจะให้ผลลัพธ์เป็นจริงหรือเท็จ อย่างใดอย่างหนึ่ง ส่วน 'a=2' จะหมายถึง_____
- การจะเลือกใช้ Relational operator ตัวใดนั้นก็ขึ้นอยู่กับความเหมาะสม อย่างในตัวอย่าง เราสามารถใช้ n < 201 แทน n<=200 ได้เช่นเดียวกัน

Program 5.3 Generating a Table of Triangular Numbers

```
// Program to generate a table of triangular numbers
#include <stdio.h>
int main (void)
{
    int n, triangularNumber;
    printf("TABLE OF TRIANGULAR NUMBERS\n\n");
    printf(" n Sum from 1 to n\n");
    printf("--- ---------------\n");
    triangularNumber = 0;
    for( n = 1; n <= 10; ++n )
    {
        triangularNumber += n;
        printf (" %i %i\n", n, triangularNumber);
    }
    return 0;
}
```

```
TABLE OF TRIANGULAR NUMBERS
n    Sum from 1 to n
---  ---------------
1    1
2    3
3    6
4    10
5    15
6    21
7    28
8    36
9    45
10   55
```

- สังเกตการใช้ ++n และ triangularNumber += n;

<u>การใช้คำสั่ง scanf เพื่อรับค่า input จากผู้ใช้</u>

Program 5.4 Asking the User for Input

```
#include <stdio.h>
int main(void)
{
    int n, number, triangularNumber;
    printf ("What triangular number do you want? ");
    scanf ("%i", &number);
    triangularNumber = 0;
    for( n = 1; n <= number; ++n )
        triangularNumber += n;
    printf ("Triangular number %i is %i\n", number,
triangularNumber);
    return 0;
}
```

```
What triangular number do you want? 100
Triangular number 100 is 5050
```

- คำสั่ง scanf รับค่าคล้ายกับ printf คือมีอักขระควบคุมรูปแบบ %i บอกว่าคำสั่งต้องการรับค่าเป็นตัวเลขจำนวนเต็ม และ เอาไปเก็บไว้ที่ตัวแปร number
- การส่งค่าตัวแปรให้ scanf จะต้องใช้ & นำหน้าชื่อตัวแปรทุกครั้ง เป็นเรื่องของ pointer ซึ่งจะกล่าวโดยละเอียดต่อไปในบทที่ 11

การใช้ for loop ซ้อนกัน

Program 5.5 Using Nested for Loops

```
#include <stdio.h>
int main (void)
{
    int n, number, triangularNumber, counter;
    for (counter = 1; counter <= 5; ++counter )
    {
        printf ("What triangular number do you want? ");
        scanf ("%i", &number);
        triangularNumber = 0;
        for ( n = 1; n <= number; ++n )
            triangularNumber += n;
        printf ("Triangular number %i is %i\n\n", number,
triangularNumber);
    }
    return 0;
}
```

```
What triangular number do you want? 12
Triangular number 12 is 78

What triangular number do you want? 25
Triangular number 25 is 325

What triangular number do you want? 50
Triangular number 50 is 1275

What triangular number do you want? 75
Triangular number 75 is 2850

What triangular number do you want? 83
Triangular number 83 is 3486
```

- ในโปรแกรมมีลูป for ซ้อนกันสองลูป โดยลูปนอกบังคับให้คำสั่งภายใต้ {...} ทำซ้ำห้าครั้ง และลูปในซึ่งทำเช่นเดียวกับโปรแกรมที่ 5.4
- ดังนั้นโปรแกรม 5.5 คือโปรแกรม 5.4 ที่รันซ้ำกันห้าครั้ง

- ลูป for สามารถซ้อนกันได้มากถึง 127 ชั้น
- สังเกตการจัดย่อหน้าในโปรแกรมซึ่งช่วยให้โปรแกรมอ่านได้ง่ายขึ้น

<u>การใช้ลูป for ในรูปแบบอื่นๆ</u>

- กรณีต้องการกำหนดค่าเริ่มต้นหลายค่า

```
for ( i = 0, j = 0; i < 10; ++i )
{
    statement;
}
```

- กรณีต้องการเปลี่ยนค่าให้กับตัวแปรหลายตัว

```
for ( i = 0, j = 100; i < 10; ++i, j = j - 10 )
{
    statement;
}
```

- กรณีไม่ต้องการกำหนดค่าเริ่มต้นเนื่องจากมีค่าเริ่มต้นกำหนดไว้ก่อนหน้าแล้ว

```
for ( ; j != 100; ++j )
{
    statement;
}
```

- กรณีที่ไม่กำหนด loop_condition หรือเงื่อนไขของลูป ลูปจะถูกรันต่อเนื่องไม่หยุด (ในกรณีนี้คำสั่ง return, break, หรือ goto อาจจะใช้เพื่อออกจากลูปได้)

```
for ( i = 0;; i++)
{
     statement;
}
```

- กรณีที่ต้องการประกาศตัวแปรพร้อมกำหนดค่า ในกรณีนี้ตัวแปรที่ประกาศจะถูกเรียกใช้ได้จากในลูป for เท่านั้น (local variable)

```
for ( int counter = 1; counter <= 5; ++counter )
{
     statement;
}
printf("counter = %i", counter);  //unknown variable
```

- ตัวอย่างอื่นๆ

```
for(i=1; i<5; ++i){…}

for(j=0; j<100; j+=20){…}

for(m=5; m>=0; m--){…}

for(n=0; n<25; n*=5){…}
```

5.2 คำสั่ง While

- รูปแบบการใช้คำสั่ง while

```
while(loop_condition)
{
    statement_1;
    statement_2;
    ...
}
```

- ก่อนที่จะเข้าลูป เงื่อนไขใน loop_condition จะถูกทดสอบ หากผลลัพธ์เป็น TRUE โปรแกรมในลูปจะถูกรันหนึ่งครั้ง หลังจากนั้น loop_condition จะถูก

ทดสอบอีกครั้ง หากเป็น TRUE โปรแกรมในลูปก็จะถูกทำซ้ำอีกครั้ง การทำงานเช่นนี้จะวนซ้ำจนกว่า loop_condition จะเป็น FALSE จึงจะหลุดออกจากลูป

Program 5.6 Introducing the while Statement

```
// Program to introduce the while statement
#include <stdio.h>
int main (void)
{
    int count = 1;
    while( count <= 5 )
    {
        printf ("%i\n", count);
        ++count;
    }
    return 0;
}
```

- ลูป while และลูป for สามารถเขียนแทนกันได้ดังนี้

```
for(init_expression; loop_condition; loop_expression)
{
    statement_1;
    statement_2;
    ...
}
```

```
init_expression;
while(loop_condition)
{
    statement_1;
    statement_2;
    ...
    loop_expression;
}
```

- การเลือกใช้งาน for หรือ while ขึ้นอยู่กับหลายปัจจัย โดยปกติหากต้องการวนลูปเป็นจำนวนครั้งที่แน่นอน จะใช้ for แต่ถ้ามีเงื่อนไขอื่นๆมาเกี่ยวข้องจะใช้ while
- ตัวอย่างโปรแกรมถัดไปเป็นการใช้ while เพื่อช่วยการคำนวณค่าหารร่วมมาก (greatest common divisor: gcd)

algorithm สำหรับการหาค่า gcd ของจำนวนเต็มค่าบวก u และ v step 1: หาก v เท่ากับ 0 ค่า gcd จะเท่ากับ v step 2: คำนวณ temp = u% v, u = v, v = temp และกลับไปทำ step 1

- จะเห็นว่าทั้งสอง step จะถูกทำซ้ำจนกว่าค่า v จะเท่ากับ 0 ดังนั้นเราใช้ while

Program 5.7 Finding the Greatest Common Divisor

```
/* Program to find the greatest common divisor
of two nonnegative integer values */
#include <stdio.h>
int main(void)
{
    int u, v, temp;
    printf("Please type in two nonnegative integers.\n");
    scanf("%i%i", &u, &v);
    while( v != 0 )
    {
        temp = u % v;
        u = v;
        v = temp;
    }
    printf ("Their greatest common divisor is %i\n", u);
    return 0;
}
```

```
Please type in two nonnegative integers.
1026 405
Their greatest common divisor is 27
```

- ตัวอย่างโปรแกรมที่ใช้ while ในการกลับตัวเลขจำนวนเต็ม เช่น หากใส่ค่า 1234 โปรแกรมจะกลับหลักของตัวเลขและแสดงค่า 4321
- ในการนี้จะต้องแยกค่าแต่ละหลักออกมาให้ได้ก่อนโดยวิธี _____

Program 5.8 Reversing the Digits of a Number

```
// Program to reverse the digits of a number
#include <stdio.h>
int main(void)
{
    int number, right_digit;
    printf("Enter your number.\n");
    scanf("%i", &number);
    while(number != 0)
    {
        right_digit = number % 10;
        printf ("%i", right_digit);
        number = number / 10;
    }
    printf ("\n");
    return 0;
}
```

```
Enter your number.
13579
97531
```

5.2 คำสั่ง do-while

- ทั้งลูป for และ while จะมีการทดสอบเงื่อนไขก่อนจะทำคำสั่งในลูปเสมอ ดังนั้น คำสั่งในลูปจะไม่ถูกทำตามเลย หากเงื่อนไขเป็น FALSE ตั้งแต่เริ่มต้น

- ในบางกรณีเราอาจต้องการให้มีการทำคำสั่งในลูปก่อนอย่างน้อยหนึ่งครั้ง จึงจะทดสอบเงื่อนไข
- สามารถทำได้โดยใช้ลูป do-while ซึ่งเขียนได้ดังนี้

```
do
{
    statement_1;
    statement_2;
    ...
}while (loop_expression);
```

- การทำงานของลูป do-while เริ่มจากกระทำคำสั่งในลูปก่อนหนึ่งรอบ จากนั้น loop_expression จะถูกทดสอบ หากเป็น TRUE คำสั่งในลูปจะถูกทำอีกครั้ง เช่นนี้ซ้ำไปเรื่อยๆ จนกว่า loop_condition จะเป็น false ลูปจึงจะหยุด
- ดังนั้นลูป do-while จึงทำงานกลับกับลูป while
- ในโปรแกรม 5.8 จะมีการทดสอบเงื่อนไขก่อนเข้าลูป หากค่าที่ใส่ให้โดยผู้ใช้เป็น 0 โปรแกรมจะไม่ทำงานคำสั่งในลูปเลย และหน้าจอจะไม่แสดงค่าผลลัพธ์ใดๆ
- เราจะแก้ไขโปรแกรมให้คำสั่งในลูปมีการทำงานก่อนอย่างน้อยหนึ่งครั้ง โดยใช้ do-while

Program 5.9 Implementing a Revised Program to Reverse the Digits of a Number

```
// Program to reverse the digits of a number
#include <stdio.h>
int main ()
{
    int number, right_digit;
    printf("Enter your number.\n");
    scanf("%i", &number);
    do
    {
        right_digit = number % 10;
        printf ("%i", right_digit);
        number = number / 10;
    }
    while ( number != 0 );
    printf ("\n");
    return 0;
}
```

5.3 คำสั่ง break

- ใช้ออกจากลูปก่อนที่เงื่อนไขใน loop_condition จะเป็นเท็จ โดยโปรแกรมจะออกจากลูปทันทีที่เจอคำสั่ง break และทำงานตามคำสั่งที่ตามหลังลูป
- หากมีหลายลูปซ้อนกัน และคำสั่ง break อยู่ในลูปในสุด โปรแกรมจะออกจากลูปในสุดเท่านั้น
- รูปแบบการใช้คำสั่ง break คือ break;
- ปกติคำสั่ง break จะใช้ร่วมกับคำสั่ง if ซึ่งจะได้กล่าวต่อไปในบทที่ 6

5.4 คำสั่ง continue

- คำสั่ง continue ใช้ข้ามคำสั่งในลูป โดยจะข้ามคำสั่งในลูปที่ตามหลังคำสั่ง continue แต่ทำลูปต่อไปตามปกติ
- รูปแบบการใช้คำสั่ง continue คือ continue;
- ปกติคำสั่ง continue จะใช้ร่วมกับคำสั่ง if ซึ่งจะได้กล่าวต่อไปในบทที่ 6

Math functions in C

- การใช้ฟังก์ชันทางคณิตศาสตร์ จะต้องมีการประกาศไลบรารี math.h ไว้ที่ header เสมอ

acos(x)	acosh(x)	asin(x)	asinh(x)	atan(x)
atanh(x)	ceil(x)	cos(r)	cosh(x)	exp(x)
fabs(x)	floor(x)	fmax(x,y)	fmin(x,y)	fmod(x,y)
log(x)	log2(x)	log10(x)	round(x)	pow(x,y)
sin(x)	sinh(x)	sqrt(x)	tan(r)	tanh(x)

- ตัวอย่างการใช้งาน

```
y = pow(x, 3);

temp = (-b + sqrt(b*b – 4*a*c))/(2*a);

printf(“sin45 = %f”, sin(45*M_PI/180));
```