บทที่ 7 การเขียนโปรแกรมแบบวนรอบทำซ้ำ (Loop)

รายวิชา สธ 113 การออกแบบโปรแกรมทางธุรกิจเบื้องต้น

อ.อภิพงศ์ ปิงยศ

Overview

- การวนรอบทำซ้ำ (Loop)
- คำสั่ง while Statement
- คำสั่ง do-while Statement
- คำสั่ง for Statement
- คำสั่ง break และ continue

การวนรอบทำซ้ำ (Loop)

- ในการเขียนโปรแกรม จะมีการประมวลผลซ้ำ (Loop หรือ Iteration) เพื่อให้โปรแกรมทำงานตาม Statement หรือการประมวลผลที่กำหนดไว้ ซ้ำกันมากกว่า 1 ครั้ง *โดยไม่จำเป็นที่จะต้องเขียน Statement นั้น ซ้ำไปซ้ำมาในโค้ด*
- การทำงานจะทำงานตาม Statement ไปจนหมด แล้วถ้าหากเงื่อนไขที่กำหนดไว้ให้ทำซ้ำยังเป็นจริง โปรแกรมจะวนกลับไปทำงานตาม Statement อีกรอบจนกว่าเงื่อนไขที่กำหนดไว้จะเป็นเท็จ จึงจะออกจากคำสั่งทำซ้ำ

ส่วนประกอบของ Iteration

มีอยู่ 3 ประเภท

- 1) **Initialization** คือ การกำหนด*ค่าเริ่มต้น*ของตัวแปรที่จะเป็นเงื่อนไขในการ iteration

  เช่น x = 1

- 2) **Testing** คือ การทดสอบว่า เงื่อนไขที่ทำการ Iteration นั้นยังเป็นจริงหรือไม่ จะมีการทำ Iteration ไปเรื่อยๆ หากเงื่อนไขยังเป็นจริง

  เช่น x < 20

- 3) **Incrementing** เป็นการเปลี่ยนแปลงค่าของตัวแปรที่ใช้เป็นเงื่อนไขในการ Iteration

  เช่น x = x + 1 หรือ x++

ประเภทของ Iteration Statement

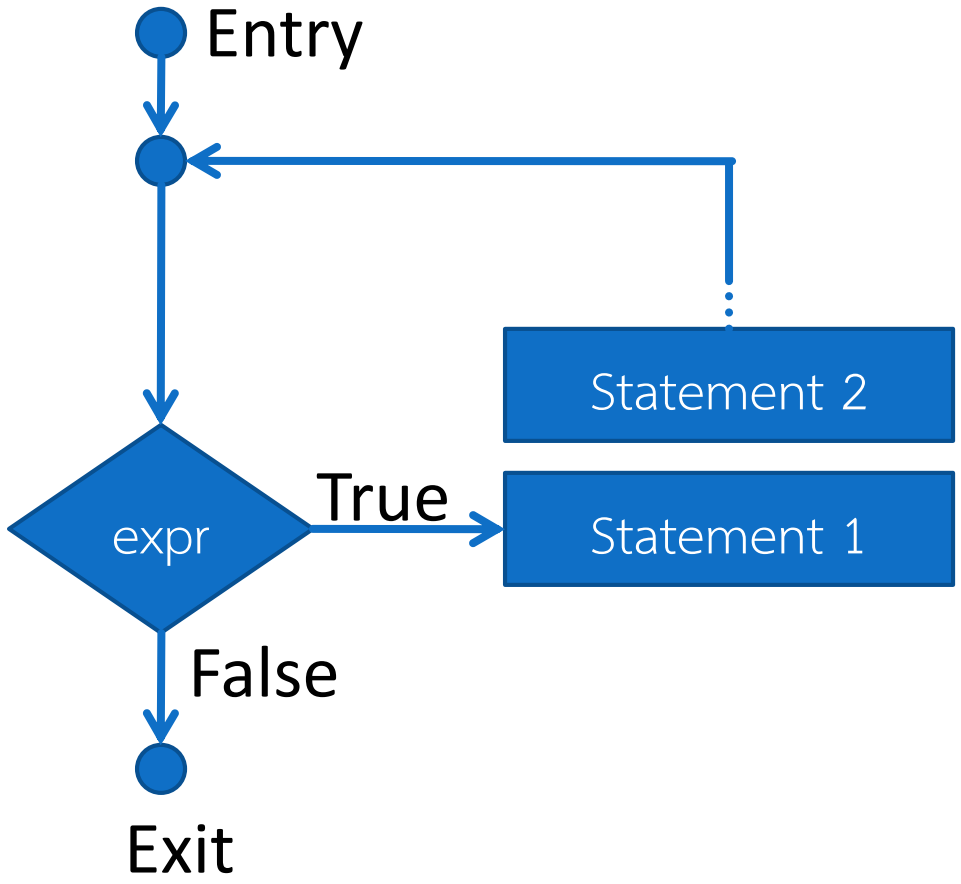
- คือ วิธีการทำให้โปรแกรมทำงานเป็น Loop ได้
- ในภาษา C มีอยู่ 4 ชนิดด้วยกัน คือ
  - **while** statement
  - **do-while** statement
  - **for** statement
  - **break** and **continue** statement

While Statement

- มีการตรวจสอบเงื่อนไขก่อน หากเงื่อนไขเป็นจริง โปรแกรมจะเริ่มทำตาม Statement ที่กำหนดไว้ แล้วกลับไปเริ่มต้นใหม่ จนกว่าเงื่อนไขจะเป็นเท็จ ถึงจะหยุดการทำงาน แล้วออกไปจาก Loop

```
while (condition)
{
   statement1;
   statement2;
   ...
   statementN;
}
```

while statement flowchart



Example 1: พิมพ์เลข 1 ถึง 10 โดยใช้ while loop

```c
#include <stdio.h>
void main()
{
    int count = 1;
    printf("Print count from 1 to 10\n");
    while (count <= 10) {
        printf("%d ", count);
        count++;
    }
    printf("\n");
}
```

```
Print count from 1 to 10
1 2 3 4 5 6 7 8 9 10
```

Example 2: เลือกการทำงานของ ATM จากตัวเลข Menu ที่กำหนดไว้ หากเลือกตัวเลขนอกเหนือที่กำหนดไว้ จะต้องใส่ตัวเลขที่เลือกใหม่ (while)

```c
#include <stdio.h>
#include <stdlib.h>
void main() {
    int input = 0;
    while(input != 1 && input != 2 && input != 3){
        printf("1. Withdraw\n");
        printf("2. Show Balance\n");
        printf("3. Transfer\n");
        printf("Please enter your choice (1-3): ");
        scanf("%d",input);     // read selection from keyboard
        switch (input) {
        case 1:
            printf("Withdraw\n");
            break;
        case 2:
            printf("Show Balance\n");
            break;
        case 3:
            printf("Transfer\n");
            break;
        default:
            printf("Please input your choice correctly\n");
        }
    }
    system("PAUSE");
}
```

Example 2: Output

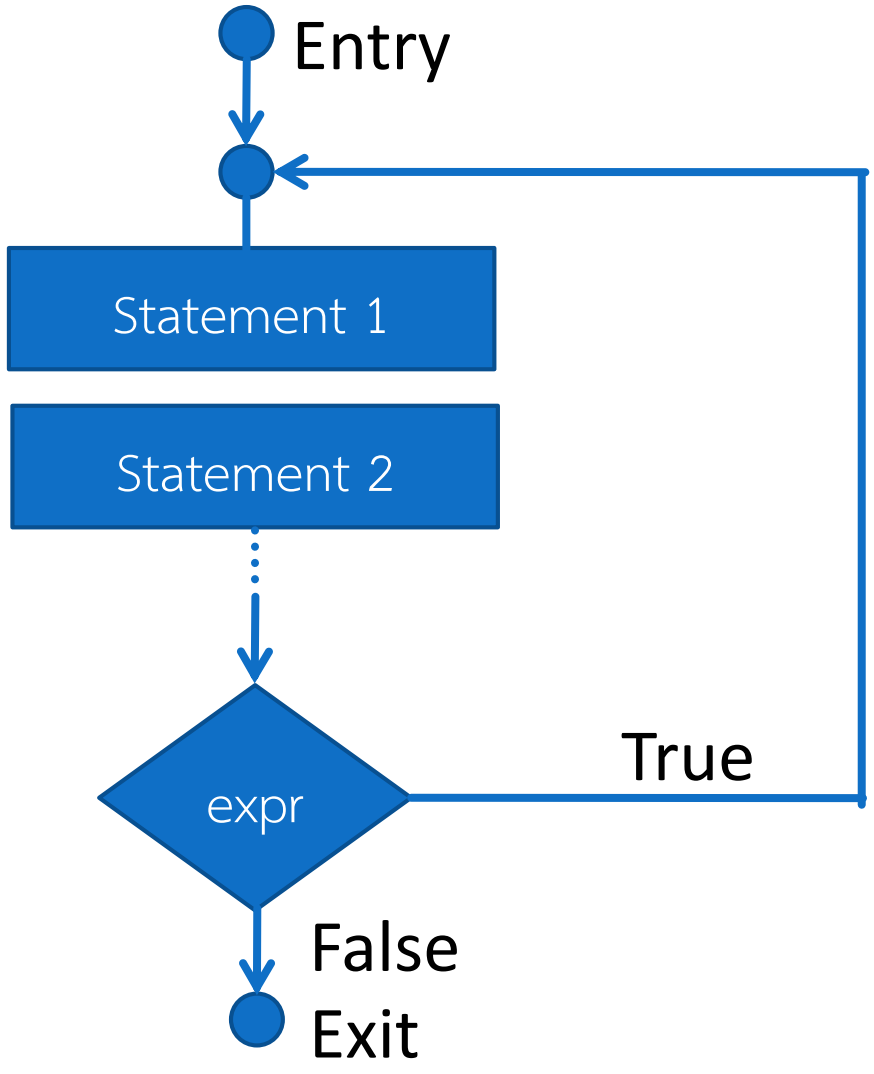
```
1. Withdraw
2. Show Balance
3. Transfer
Enter your choice (1-3): 5
Please input your choice correctly.
1. Withdraw
2. Show Balance
3. Transfer
Enter your choice (1-3): 1
Withdraw
```

do while Statement

- จะแตกต่างจาก while Statement ตรงที่โปรแกรมจะเริ่มทำงานตาม Statement ที่ระบุไว้ โดยไม่มีการเช็คเงื่อนไขก่อน จากนั้นเมื่อทำงานเสร็จแล้วหนึ่งรอบถึงจะเช็คเงื่อนไข หากเงื่อนไขเป็นจริง จะวนกลับไปทำซ้ำ แต่หากเงื่อนไขเป็นเท็จจะจบการทำงานแล้วออกจาก Loop ทันที

```
do {
   statement1;
   statement2;
   ...
   statementN;
} while (condition);
```

do while statement flowchart



Example 3: พิมพ์เลข 1 ถึง 10 โดยใช้ do while loop

```c
#include <stdio.h>
void main() {
    int count;
    count = 1;
    printf("Print count from 1 to 10\n");
    do {
        printf("%d ", count);
        count++;
    } while (count <=10);
    printf("\n");
}
```

```
Print count from 1 to 10
1 2 3 4 5 6 7 8 9 10
```

Example 4: เลือกการทำงานของ ATM จากตัวเลข Menu ที่กำหนดไว้ หากเลือกตัวเลขนอกเหนือที่กำหนดไว้ จะต้องใส่ตัวเลขที่เลือกใหม่ (do-while)

```c
void main() {
    int input;
    do{
        printf("1. Withdraw\n");
        printf("2. Show Balance\n");
        printf("3. Transfer\n");
        printf("Please enter your choice (1-3): ");
        scanf("%d",input);     // read selection from keyboard
        switch (input) {
        case 1:
            printf("Withdraw\n");
            break;
        case 2:
            printf("Show Balance\n");
            break;
        case 3:
            printf("Transfer\n");
            break;
        default:
            printf("Please input your choice correctly\n");
        }
    }while (input != 1 && input != 2 && input != 3);
    system("PAUSE");
}
```

Example 4: Output

```
1. Withdraw
2. Show Balance
3. Transfer
Enter your choice (1-3): 4
Please input your choice correctly.
1. Withdraw
2. Show Balance
3. Transfer
Enter your choice (1-3): 2
Show Balance
```

for Statement

- ในการเขียนโปรแกรม เมื่อต้องการให้มีการประมวลผลซ้ำ (Loop) โดยที่เรา*ทราบจำนวนของการทำซ้ำ* เราสามารถใช้ for Loop แทน while Loop ได้
- เช่น ทำการคำนวณค่าเฉลี่ยของกลุ่มตัวเลขจำนวน 10 ค่า

แต่ก็ยังสามารถใช้งาน while หรือ do while ได้อยู่เช่นกัน

for statement syntax

```
for (initial; condition; incrementing)  {
    statement 1;
    statement 2;
    …
    statement n;
}
```

เมื่อ

initial: การกำหนดค่าเริ่มต้นตัวแปรที่ใช้ในการคุมการวนรอบ

condition: เงื่อนไขของการวนรอบ

incrementing: การเปลี่ยนแปลงค่าของตัวแปรควบคุมแต่ละรอบ

for จะแตกต่างจาก while และ do while ที่
ใน while(condition) จะมีเพียงเงื่อนไขที่จะทำซ้ำเท่านั้น
ส่วนค่าเริ่มต้นและการเปลี่ยนแปลงค่าของตัวแปรควบคุม
การทำซ้ำจะอยู่ที่อื่น เช่นการกำหนดค่าเริ่มต้นจะอยู่
ภายนอกก่อนเข้า Loop หรือ การเปลี่ยนแปลงค่าของตัว
แปรอยู่ภายใน Statement ที่อยู่ในลูป เป็นต้น

for statement syntax [cont.]

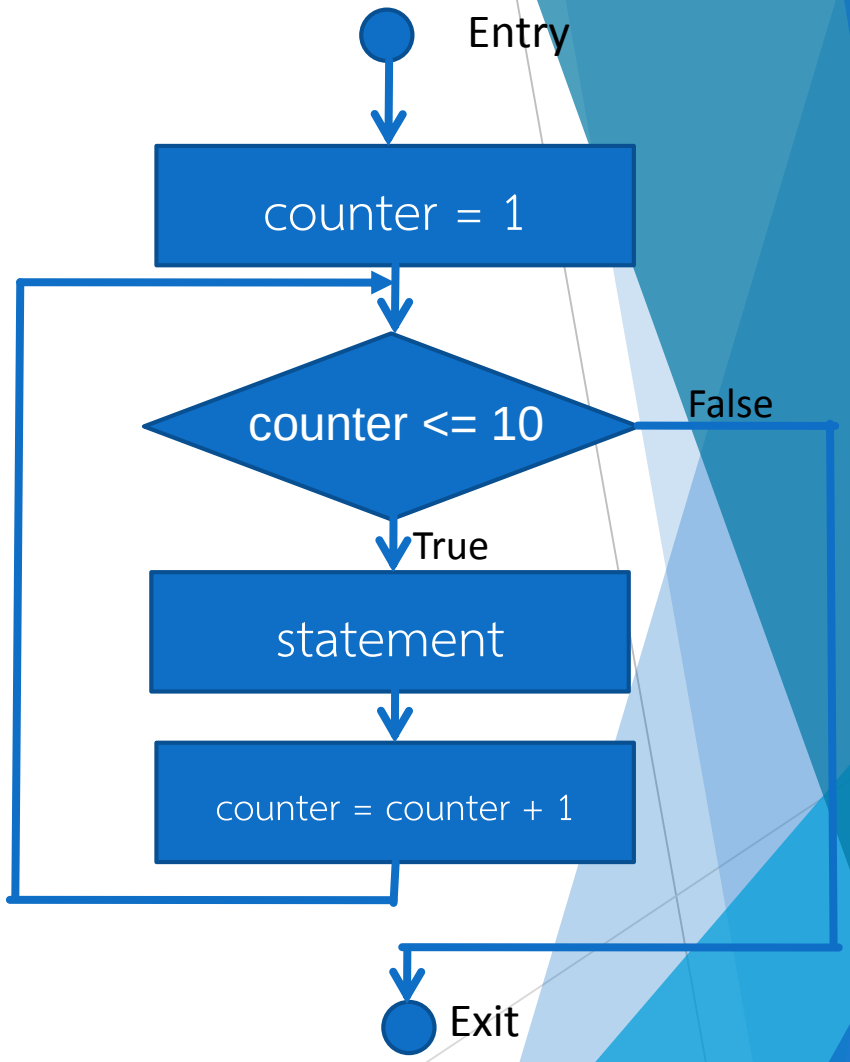
- สามารถเปรียบเทียบกับการใช้ while ได้ดังนี้

```
initial;
while (condition)  {
   statement1;
   …
   statementn;
   incrementing;
}
```

for statement flowchart

- ตัวอย่าง สำหรับ

```
for (counter = 1; counter <=10; counter = counter+1) {
    statement;
}
```



Example 5: พิมพ์เลข 1 ถึง 10 โดยใช้ for loop

```c
#include <stdio.h>
int main() {
    int counter;
    printf("Print counter from 1 to 10\n");
    for (counter = 1; counter <=10; counter = counter + 1) {
        printf("%d ", counter);
    }
    return 0;
}
```

```
Print count from 1 to 10
1 2 3 4 5 6 7 8 9 10
```

ตัวดำเนินการสำหรับกำหนดค่าแบบย่อ

- ในการเพิ่มค่าของตัวแปรควบคุมแต่ละรอบ หรือส่วนสุดท้ายของคำสั่งใน while, do while, for เราสามารถใช้ตัวดำเนินการสำหรับกำหนดค่า (Assignment Operator) แบบย่อได้

ตัวดำเนินการ	ตัวอย่างการใช้งาน	ความหมาย	ค่าที่ได้
+=	c += 7	c = c + 7	c เท่ากับ 10
-=	d -= 4	d = d – 4	d เท่ากับ 1
***=**	e *= 5	e = e * 5	e เท่ากับ 20
/=	f /= 3	f = f / 3	f เท่ากับ 2
%=	g %= 9	g = g % 9	g เท่ากับ 3

เมื่อกำหนดให้ c = 3, d = 5, e = 4, f = 6, g = 12

ตัวดำเนินการสำหรับกำหนดค่าแบบย่อ [cont.]

- นอกการจากนั้น ยังมีตัวดำเนินการ ++ และ – สำหรับเพิ่มและลดค่าทีละ 1 การใช้งานมีสองแบบ คือ วางข้างหน้า และ วางข้างหลังตัวแปร หากวางข้างหน้าตัวแปร ตัวแปรจะถูกเพิ่มหรือลดค่าก่อน จะถูกนำค่าไปใช้งาน หากวางข้างหลังตัวแปร ตัวแปรจะถูกค่าไปใช้งานก่อน แล้วจึงถูกเพิ่มหรือลดค่า

ตัวดำเนินการ	ตัวอย่างการใช้งาน	ความหมาย
++	++a	เพิ่มค่า a ขึ้น 1 แล้วจึงนำค่า a ไปใช้
++	a++	ใช้ค่า a ปัจจุบัน แล้วเพิ่มค่า a ขึ้น 1
--	--b	ลดค่า b ลง 1 แล้วจึงนำค่า b ไปใช้
--	b--	ใช้ค่า b ปัจจุบัน แล้วลดค่า b ลง 1

คำสั่ง break และ continue

- break และ continue ใช้ในการเปลี่ยนแปลงเส้นทางการทำงานของโปรแกรม
- แต่หากเป็นไปได้ ควรหลีกเลี่ยงการใช้งาน break และ continue เพื่อให้โปรแกรมมีโครงสร้างที่ดี

break statement

- การใช้งาน break ในประโยคคำสั่งทำซ้ำ while, do while, for หรือ switch จะทำให้ออกจากประโยคคำสั่งทำซ้ำทันที
- จะใช้ break เพื่อให้โปรแกรมกระโดดข้ามไปทำงานในส่วนที่เหลือที่อยู่นอก Loop ทันที

Example 6: เป็นการเขียนโปรแกรม แสดง counter โดยใช้ for loop หาก counter มีค่าเท่ากับ 5 จะต้องออกจาก for loop ทันที

```
2  void main() {
3      int x;
4      for (x = 1; x <= 10; x++) {
5          if (x == 5) {
6              /* if x is 5, terminate loop */
7              break;
8          }
9          printf("%d", x);
10     }
11     printf("\nBroke out of loop at x = %d\n", x);
12 }
13
```

```
1 2 3 4
Broke out of loop at x = 5
```

continue statement

- การใช้งาน continue ในประโยคคำสั่งทำซ้ำ while, do while, for จะมีผลทำให้การทำงานกระโดดข้ามคำสั่งที่เหลืออยู่ใน Loop แล้วกลับไปเริ่มทำงานในรอบต่อไปใหม่ทันที
  - สำหรับ while และ do while เงื่อนไขของการวนรอบ จะถูกทดสอบทันที
  - สำหรับ for ตัวแปรนับจะถูก ลด/เพิ่ม ค่า แล้วทำการทดสอบเงื่อนไขการวนรอบ

Example 7: เป็นการเขียนโปรแกรมแสดง counter โดยใช้ for loop แต่เมื่อ counter มีค่าเท่ากับ 5 จะให้กระโดดข้ามคำสั่ง printf ออกไป แล้วไปเพิ่ม counter จากการทำงานของ for loop ทันที

```c
#include <stdio.h>
int main() {
    int x;
    for (x = 1; x <= 10; x++) {
        if (x == 5) {
            /* if x is 5, skip remaining code in loop body*/
            continue;
        }
        printf("%d", x);
    }
    printf ("\n Used continue to skip printing the value 5\n");
    return (0);
}
```

```
1 2 3 4 6 7 8 9 10
Used continue to skip printing the value 5
```