w09-Lab

Iterative Control Statements Part II

Adapted for 204111

by Areerat Trongratsameethong

Topics

- **Basic Loop Structures (Revisited)**
- **Pretest Loop/Posttest Loop (Revisited)**
- **The while Statement (Revisited)**
- **Sentinels**
- **Break and Continue**
- **The for Statement**
- **Nested Loop**
- **The do while Statement**

Basic Loop Structures [Revisited]

การสร้าง Loop ประกอบด้วย 4 องค์ประกอบหลักคือ

1. **Repetition statement: คำสั่งที่ใช้สำหรับการทำซ้ำ**
   - ☑ `while statement`
   - ☐ `for statement`
   - ☐ `do-while statement`
2. **Condition: เงื่อนไขของการทำซ้ำ**
3. **A statement that initially sets the condition being tested: การตั้งค่าเริ่มต้นให้กับตัวแปรที่ใช้ในการควบคุมเงื่อนไข**
4. **A statement within the repeating section of code that alters the condition so that it eventually becomes false: การเปลี่ยนแปลงค่าตัวแปรที่ใช้ในการควบคุมเงื่อนไข ในการวนแต่ละครั้ง เพื่อให้เงื่อนไขเป็นเท็จในที่สุด**

Pretest Loop

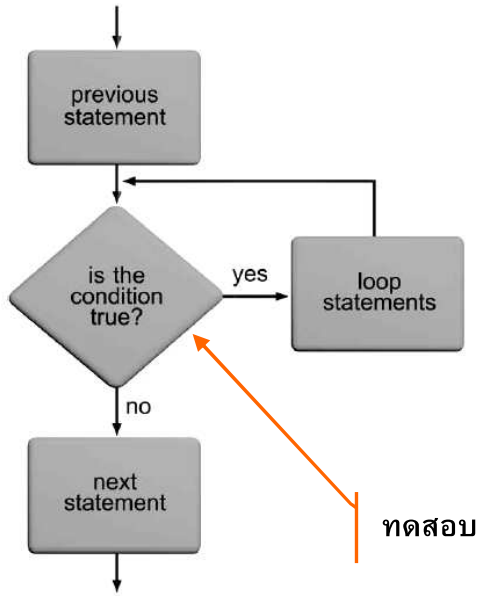
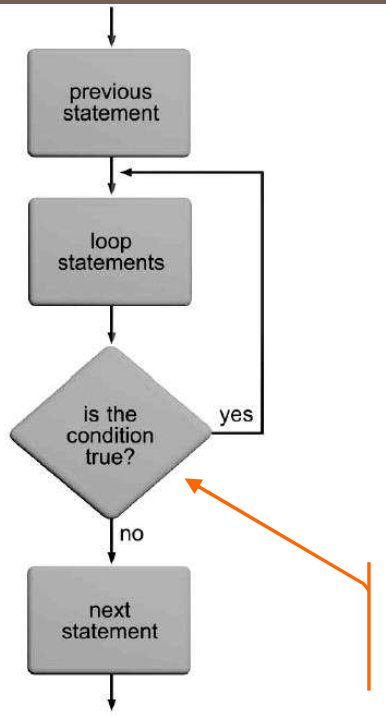


Figure 5.1 A pretest (entrance-controlled) loop

ทดสอบเงื่อนไขก่อนเข้าทำใน Loop

Posttest Loop



ทดสอบเงื่อนไขหลังจากเข้า Loop ดังนั้น Loop ลักษณะนี้จะถูกดำเนินการอย่างน้อย 1 ครั้ง

Figure 5.2 A posttest (exit-controlled) loop

The while Statement

```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 int main()
05 {
06     int count;       /* initialize count */
07     count = 1;
08
09     while (count <= 10) {
10         printf("%d", count);
11         count++;     /* add 1 to count */
12     }
13
14     printf("\n");    /*print a blank line */
15
16     return 0;
17 }
```

Output is:
1 2 3 4 5 6 7 8 9 10

4 องค์ประกอบที่สำคัญ

Sentinels

- การระบุจำนวนครั้งที่ทำซ้ำไว้เป็นค่าคงที่ ในบางครั้งอาจไม่ยืดหยุ่นเพียงพอกับปัญหาที่ต้องการแก้
  - เช่นกรณีต้องการหาค่าเฉลี่ยของคะแนนของนักเรียนในชั้น แต่ไม่ทราบจำนวนนักเรียนล่วงหน้า
- สามารถแก้ปัญหาได้โดย
  - ให้ user ระบุจำนวนครั้งที่ต้องการทำซ้ำ ผ่าน input
  - หรือ อ่าน input จาก user จนเจอค่าที่ตกลงกันไว้ว่าใช้แสดงจุดสิ้นสุดของข้อมูล เช่น -1 หรือ EOF (End of File)
    - ค่าที่ใช้แสดงจุดเริ่มหรือสิ้นสุดของข้อมูลในลักษณะนี้เรียกว่า *sentinels*
    - ควรเลือกค่า sentinels ให้ไม่ทับซ้อนกับค่าของข้อมูลที่เป็นไปได้

Sentinels (2)

```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 #define SENTINEL -1
05 int main()
06 {
07
08     int count;
09     double score, total, average;
10
11     printf("Please enter students' score one for each line\n"
12     printf("and %d for termination: \n",SENTINEL);
13     total = 0.0;
14     count = 0;
15     scanf("%le", &score);
16     while (score != SENTINEL) {
17         total += score;
18         count++;
19         scanf("%le", &score);
20     }
21     average = total / count;
22     printf("\nThe average of the %d numbers is %8.4f\n", count, average);
23
24     return 0;
25 }
```

The break Statement

- เราสามารถใช้คำสั่ง **break** เพื่อออกจาก loop ได้ตามเงื่อนไขที่ระบุ โดยคำสั่งอื่น ๆ ภายใน loop หลังจาก **break** จะถูกข้ามไป
- ใช้ได้กับคำสั่ง **for**, **while**, **do-while** และ **switch**

The break Statement (2)

```
while (count <= 10) {
    printf("Enter a number: ");
    scanf("%le", &num);
    if (num > 76)
    {
        printf("You lose!");
        break; /* break out of the loop */
    }
    else
    printf("Keep on truckin!");
}
/* break jumps to here */
```

เมื่อ num มีค่า > 76
- พิมพ์ "You lose!"
- ออกจาก while loop

The continue Statement

- คำสั่ง **continue** ใช้ได้กับ loop เท่านั้นโดยจะข้ามคำสั่งที่เหลือภายใน loop หลังจากคำสั่ง **continue** เพื่อไปวน loop ในรอบถัดไป

```
while (count < 30) {
    printf("Enter a grade: ");
    scanf("%f", &grade);

    if(grade < 0 || grade > 100) {
        continue;
    }
    total = total + grade;
    count = count + 1;
}
```

Jump to

ถ้าจริง (ไม่นับ ไม่บวก)

The for Statement

- คำสั่ง **for** เป็นคำสั่งที่รวมองค์ประกอบของ loop ทั้ง 4 ข้อไว้ในบรรทัดเดียว

```
for (กำหนดค่าเริ่มต้น; เงื่อนไขการทำซ้ำ; เปลี่ยนค่าตัวแปรควบคุม) {
  statement;
}
```

- เราไม่จำเป็นต้องระบุทั้ง 3 ค่าภายใน บรรทัดที่มีคำสั่ง **for** (เช่นสามารถกำหนด ค่าเริ่มต้นที่บรรทัดอื่น ก่อนคำสั่ง **for** ได้)
  - แต่จำเป็นต้องใส่ **semi-colon** ให้ครบทั้งสองอันภายในวงเล็บ
    - `for ( ; count <= 20; )  // valid syntax`
    - `for ( ; ; )  // also valid syntax`

การละเงื่อนไขการทำซ้ำ ทำให้เป็น infinite loop.. วิธีแก้?

Sequence of Execution

0 ทำครั้งเดียว ก่อน test เงื่อนไขครั้งแรกสุด

3 ทำทุกครั้งที่วน loop หลัง loop_statement_N ก่อน test เงื่อนไขทำซ้ำ ครั้งถัดไป

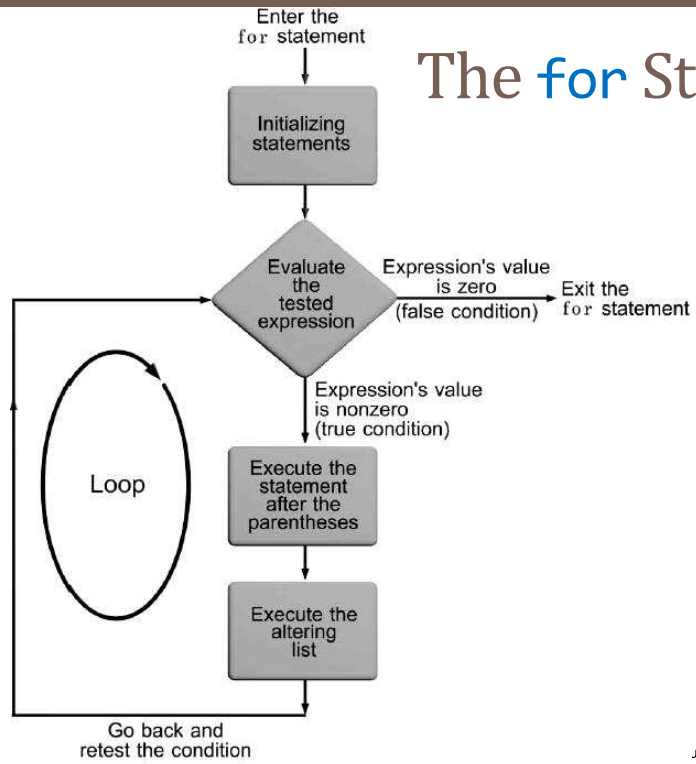
```
for (กำหนดค่าเริ่มต้น; เงื่อนไขการทำซ้ำ; เปลี่ยนค่าตัวแปรควบคุม)
{
    loop_statement_1
    loop_statement_2
    loop_statement_3


    loop_statement_N
}
```

1 test เงื่อนไขทุกครั้งที่วน loop ก่อน loop_statement_1

2

The for Statement (2)



The for Statement (3)

```c
#include <stdio.h>
#define MAXCOUNT 20

int main()
{
    int count;

    for (count = 2; count <= MAXCOUNT; count += 2) {
        printf("%d ",count);
    }

    return 0;
}
```

Output is:
2 4 6 8 10 12 14 16 18 20

The for Statement (4)

```c
01 #include <stdio.h>
02
03 int main()
04 {
05   int count;
06   count = 2; /* initializer outside for statement */
07
08   for ( ; count <= 20; count += 2) {
09     printf("%d ",count);
10   }
11
12   return 0;
13 }
```

The for Statement (5)

```
01 #include <stdio.h>
02
03 int main()
04 {
05   int count;
06   count = 2; /* initializer outside for statement */
07
08   for( ; count <= 20; ) {
09
10     printf("%d ",count);
11     count += 2; /* alteration statement */
12   }
13
14   return 0;
15 }
```


The for Statement (6)

```
01 #include <stdio.h>
02
03 int main() /* all expressions within the for's parentheses */
04 {
05   int count;
06
07   for (count = 2; count <= 20; printf("%d ",count), count += 2);
08
09   return 0;
10 }
```

Comma-separated list

Not Recommended

(Hard to Read)

The for Statement (7)

```
#include <stdio.h>
#define TABLESIZE 10
int main()
{
  int num;

  printf("NUMBER SQUARE CUBE\n");
  printf("------ ------ ----\n");

  for (num = 1; num <= TABLESIZE; num++) {
    printf("%3d %7d %6d\n",
    num, num*num, num*num*num);
  }

  return 0;
}
```

```
num = 1;

while (num <= TABLESIZE) {

    printf ("%3d %7d %6d\n",
    num, num*num, num*num*num);
    num++; /* add 1 to num */
}
```

for vs while

Practice 1

- ให้เขียนโปรแกรม โดยใช้ **for loop** เพื่อตรวจสอบค่า integer x แล้วแสดงค่า **True** เมื่อ x เป็น perfect cube และ **False** ในกรณี x ไม่เป็น perfect cube ในกรณี x ไม่เป็น perfect cube (กำลังสาม)
  - โดยให้พิจารณาว่า integer x ใด ๆ จะเป็น perfect cube ก็ต่อเมื่อมี integer y ที่ $x = y^3$ ดังนั้นหากอ่านค่า 27 จะแสดง ค่า True และหากอ่านค่า 16 จะแสดงค่า False

ตัวอย่างการ Run โปรแกรม 1

```
Enter a number: 27
True
```

ตัวอย่างการ Run โปรแกรม 2

```
Enter a number: 16
False
```

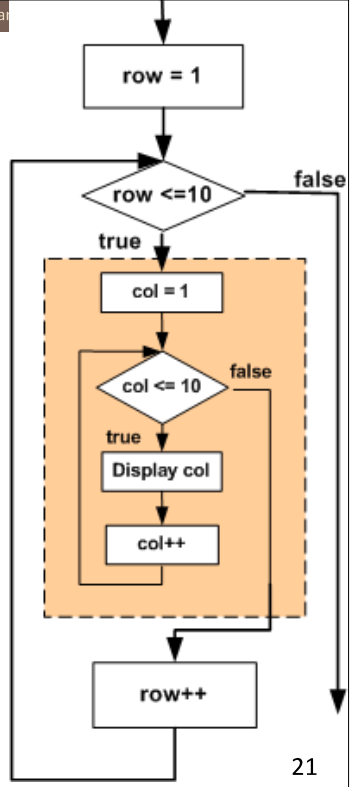
Nested Loop

Pseudocode

Initial Condition: row = 1
WHILE row <= 10
  Initial Condition: col = 1
  WHILE col <= 10
    Display col
    Add 1 to col
  ENDWHILE
  Add 1 to row
ENDWHILE

Inner Loop

Outer Loop



Reference:
http://facweb.northseattle.edu/voffenba/class/CSC110-W08/NotesDL/wk08/NoteOnNestedLoops.htm

Nested Loop (2)

```
01 #include <stdio.h>
02 #include <math.h>
03
04 #define ROW 5
05 #define COLUMN 4
06
07 int main()
08 {
09
10     int i , j;
11
12     for (i = 1; i <= ROW; i++) {          //loop นอก starts
13         printf("\ni is now %d\n",i);
14         for (j = 1; j <= COLUMN; j++ ) {      //loop ใน starts
15             printf("j = %d ",j);
16         }                                     //loop ใน ends
17     }                                     //loop นอก ends
18
19     return 0;
20 }
```

```
i is now 1
j = 1 j = 2 j = 3 j = 4
i is now 2
j = 1 j = 2 j = 3 j = 4
i is now 3
j = 1 j = 2 j = 3 j = 4
i is now 4
j = 1 j = 2 j = 3 j = 4
i is now 5
j = 1 j = 2 j = 3 j = 4
```

A First Book of ANSI C, Fourth Edition

Nested Loops (3)

```
i = 1
   j = 1 j = 2 j = 3 j = 4
i = 2
   j = 1 j = 2 j = 3 j = 4
i = 3
   j = 1 j = 2 j = 3 j = 4
i = 4
   j = 1 j = 2 j = 3 j = 4
i = 5
   j = 1 j = 2 j = 3 j = 4
```

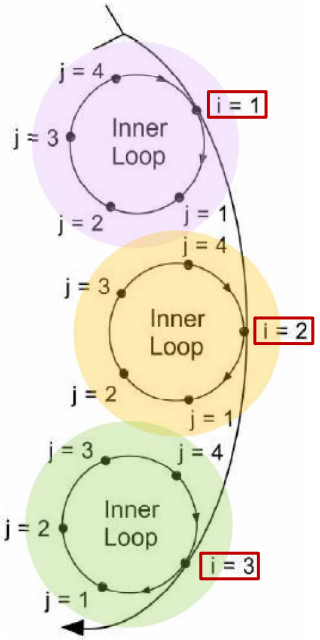


Figure 5.9 j loops once for each i

A First Book of ANSI C, Fourth Edition

Example 1

```
#include <stdio.h>

#define SIZE 5

int main()
{

    int i , j;

    for (i = 1; i <= SIZE; i++) {          //outer loop starts

        for (j = 1; j <= i; j++ ) {            //inner loop starts
            printf("%d ",j);
        }                                      //inner loop ends
        printf("\n");
    }                                      //outer loop ends

    return 0;
}
```

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

A First Book of ANSI C, Fourth Edition

Example 2

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

```c
#include <stdio.h>

#define SIZE 5

int main()
{

    int i , j;

    for (i = 1; i <= SIZE; i++) {              //outer loop starts

        for (j = 1; j <= SIZE - i + 1; j++ ) {     //inner loop starts
            printf("%d ",j);
        }                                          //inner loop ends
       printf("\n");
    }                                         //outer loop ends

    return 0;
}
```


Example 3

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

```c
#include <stdio.h>
#define SIZE 5

int main()
{

    int i, j, k;

    for (i = 1; i <= SIZE; i++) {
        for (j = 1; j <= SIZE -i+1; j++ ) {
            printf(" ");
        }

        for (k = 1; k <= i; k++ ) {
            printf("* ");
        }
        printf("\n");
    }

    return 0;
}
```


The `do-while` Statement

- **Post test loop**
  - เหมาะกับการตรวจสอบข้อมูลที่รับเข้ามา

```c
do {
    printf("Input score (-1 to terminate): ");
    scanf("%le",&score);

} while (score != -1);
```


The `do-while` Statement [2]

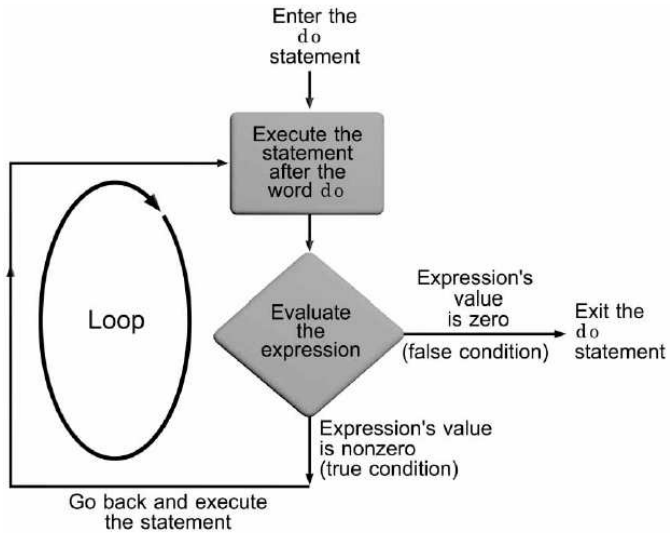


Figure 5.10 `do-while` statement flow of control

Practice 3

ให้เขียนโปรแกรมเพื่อรับค่าจำนวนเต็มที่ประกอบด้วย 1 และ 0 ไม่เกิน 8 หลัก พร้อมแสดงผลให้อยู่ในรูป เลขฐาน 2 จำนวน 8 bit โดยให้ใส่ 0 ในจำนวนหลักที่ว่าง

- ตัวอย่างการ run 1

```
Input binary: 1101
|0000 1101| is 13 in decimal
```

- ตัวอย่างการ run 2

```
Input binary: 1102
Invalid input
Input binary: 11
|0000 0011| is 3 in decimal
```


Practice 4

ให้เขียนโปรแกรมเพื่อรับค่าจำนวนเต็ม (-, 0, +) และแสดงค่าจำนวนที่กลับหลักแล้ว ดังผลการ run ด้านล่าง

- ตัวอย่างการ run 1

```
Input an integer: 94857
75849
```

- ตัวอย่างการ run 2

```
Input an integer: -1
-1
```


Practice 5

ให้เขียนโปรแกรมเพื่อรับเลขจำนวนเต็ม x แล้วตรวจสอบว่า x เป็นจำนวนเฉพาะหรือไม่ โดยใช้ for loop ดังตัวอย่างการ run ด้านล่าง

- ตัวอย่างการ run 1

```
Input an integer: 1223
1223 is prime
```

- ตัวอย่างการ run 2

```
Input an integer: 1227
1227 is NOT prime
```

หมายเหตุ: จำนวนแรกที่หาร x ลงตัวหาก x ไม่ใช่จำนวนเฉพาะจะมีค่าไม่เกิน $\sqrt{x}$


```
#include <math.h>
```

The math.h library

- เราสามารถเรียกใช้ function ทางคณิตศาสตร์ได้จาก library math.h ในปฏิบัติการนี้จะกล่าวถึงเฉพาะบาง function ที่พบบ่อย
- `double pow(double x, double y)`
  - Returns x^y
- `double sqrt(double x)`
  - Returns $\sqrt{x}$
- `double ceil(double x)`
  - Returns $\lceil x \rceil$
- `double floor(double x)`
  - Returns $\lfloor x \rfloor$
- `double fab(double x)`
  - Returns $|x|$

http://www.tutorialspoint.com/c_standard_library/math_h.htm

Practice 2

ให้เขียนโปรแกรมเพื่อคำนวณเลข fraction ในฐาน 2 โดยรับค่าเฉพาะส่วนที่อยู่หลังจุด binary point ไม่เกิน 8 ตำแหน่ง พร้อมแสดงผลเป็นทศนิยม 8 ตำแหน่ง ดังผลการ run ด้านล่าง

```
Input binary fraction: 1101
0.1101 is the sum of
1/2 +
1/4 +
0/8 +
1/16
The result is 0.81250000
```


Practice 6

ให้นำเข้าข้อมูลประเภทจำนวนเต็ม 2 ค่า คือค่าแรก (first) และค่าสุดท้าย (last) ผ่านทาง keyboard และนับว่ามีจำนวนเฉพาะกี่จำนวน อะไรบ้าง

- ตัวอย่างการ run

```
Input first integer: 3
Input last integer: 20
Prime numbers between 3 and 20 are:
3 5 7 11 13 17 19
Total: 7
```