บทที่ 4-2

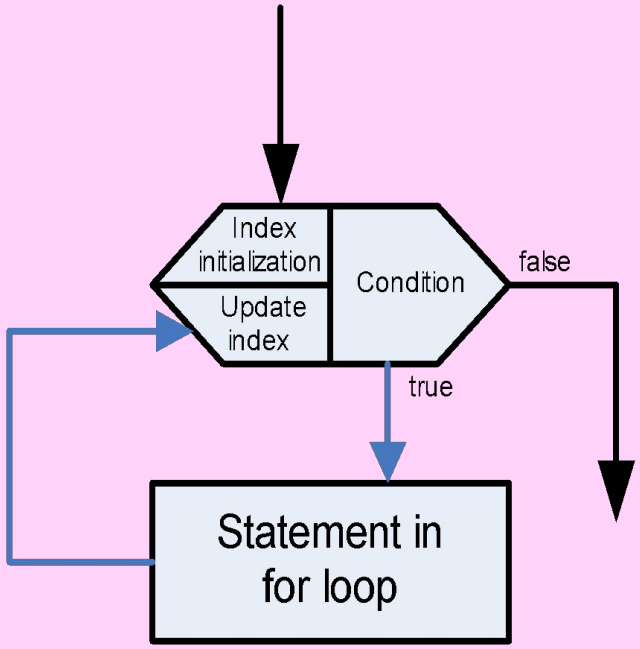
การควบคุมทิศทางการทำงานของโปรแกรม
การทำซ้ำ (Repetition)

Repetition in C Programming

- **ภาษา C มีคำสั่งเพื่อใช้สำหรับควบคุมการทำซ้ำ ดังนี้**
  - คำสั่ง for
  - คำสั่ง while
  - คำสั่ง do-while

- **วงรอบ (Loop)** ในภาษาการโปรแกรมคอมพิวเตอร์ หมายถึง ลำดับของคำสั่งในภาษาโปรแกรมที่ทำงานซ้ำต่อเนื่อง จนกว่าเงื่อนไขในการทำซ้ำนั้นจะสิ้นสุดลง

คำสั่ง for



โครงสร้างของคำสั่ง for

- **Index initialization** คือ การกำหนดค่าเริ่มต้นให้กับตัวแปรดัชนี

  ตัวแปรดัชนี (index) คือ ตัวที่ใช้ควบคุมการทำงานในวงรอบ

- **Update index** คือ การเพิ่มค่า หรือลดค่า index เพื่อให้ผลของการเปรียบเทียบเงื่อนไข (condition) มีแนวโน้มที่จะเป็นเท็จได้ เช่น **n++** หรือ **n--**
- **Condition** คือ เงื่อนไขในการเปรียบเทียบ เช่น **n <= 9**

คำสั่ง for

- ไวยากรณ์ของคำสั่ง for

  for (index initialization; Condition; Update Index)

  Statement ;

ตัวอย่างเช่น

```
for (n = 1; n <= 10; n++)
    printf("RMUTT\n");
```

ผลลัพธ์ของ Loop นี้จะแสดงคำว่า RMUTT ทั้งหมด 10 ครั้ง (รอบที่ 1-10)

หรือ

```
for (n = 10; n >= 1; n--)
    printf("RMUTT\n");
```

ผลลัพธ์ของ Loop นี้จะแสดงคำว่า RMUTT ทั้งหมด 10 ครั้ง (รอบที่ 10-1)

คำสั่ง for

- คำสั่ง for สามารถทำซ้ำได้เพียงหนึ่งคำสั่งเท่านั้น ถ้าต้องการให้ทำซ้ำมากกว่าหนึ่งคำสั่งต้องใส่คำสั่งเหล่านั้นไว้ภายใต้ Compound Statement โดยใช้เครื่องหมาย { }

```
for (n = 0; n <= 9; n++)
        {
     printf("Hello");
     printf("World");
  }
```

- ในทางเดียวกัน ถ้าใส่เครื่องหมาย ; หลังเครื่องหมายวงเล็บ) ของ for ดังตัวอย่าง

```
for (n=0; n<=9; n++) ;
```

จะทำให้ไม่เกิดการทำซ้ำคำสั่งต่อจากนี้ เนื่องจาก ; หมายถึง จบหนึ่งคำสั่ง ดังนั้น คำสั่งที่ถูกทำซ้ำคือ คำสั่งที่อยู่ก่อนหน้า ; ซึ่งก็คือ ไม่มีคำสั่งใด ๆ นั่นเอง

คำสั่ง for

- สามารถประกาศตัวแปรในส่วนของ index initialization ได้หลายตัว เช่น

```
for (int x=0,y=0; x+y<30; ++x, y+=5)
    printf("%d+%d = %d\n", x, y, x+y);
```

**ในที่นี้ Index มี 2 ตัวคือ ตัวแปร x และตัวแปร y **

```
0+0  = 0
1+5  = 6
2+10 = 12
3+15 = 18
4+20 = 24
```

จากตัวอย่าง แสดงให้เห็นว่า ในส่วนของ index initialization สามารถมีตัวแปรดัชนี (index) ได้มากกว่า 1 ตัว และในส่วนของ condition ก็สามารถใส่นิพจน์การคำนวณได้ด้วย

การประยุกต์ใช้งาน Nested for

- ตัวอย่างการแสดงสูตรคูณแม่ 1 ถึงสูตรคูณแม่ 12

```
for (int m = 1; m <= 12; m++)
{
   for (int n = 1; n <= 12; n++)
      printf("%dx%d = %d\n",m,n,m*n);
}
```

1x1 = 1

1x2 = 2

.

.

1x12 = 12

<u>*มุมขบคิด*</u>

ถ้าต้องการให้โปรแกรมนี้มีการแสดงทีละบรรทัดตามการกด enter จะต้องเปลี่ยนโปรแกรมให้เป็นแบบใด

ตัวอย่างการใช้งาน for

```
1  #include <stdio.h>
2  #include <conio.h>
3  int main(void)
4  {
5    char c;
6    for (c=getche( ); c!='q'; c=getche(
7  ));
8    printf("\n q has been found \n");
9         return(0);
   }
```

การทำงาน

โปรแกรมจะทำการวนรอบเพื่อรับค่าตัวอักษรเรื่อย ๆ จนกว่าตัวอักษรที่รับเข้ามาเป็นอักขระ ‘q’ จึงจะจบการวนรอบ

ผลลัพธ์การทำงาน

123abcq

q has been found

โปรแกรมนี้แสดงให้เห็นถึง การกำหนดค่าให้กับตัวแปรดัชนี (index) อีกรูปแบบหนึ่ง โดยการใช้ฟังก์ชัน getche( ) และมีการ update ค่าด้วยวิธีการรับค่า มิใช่การเพิ่มหรือลดค่า index

ตัวอย่างการใช้งาน for

```
1   #include <stdio.h>
2   int main(void)
3   {
4   int i;
5   for (i=1;  i<100;  i++)
6     { if ((i%2)== 0)
7         continue;
8       printf("This is odd number (%d)\n", i);
9       if (i>5)  break;
10    }
11  return(0);
12  }
```

ผลลัพธ์การทำงาน

This is odd number (1)

This is odd number (3)

This is odd number (5)

This is odd number (7)

หมายเหตุ

- คำสั่ง continue ทำหน้าที่กระโดดไปยังจุด update (i++) ทันทีโดยไม่สนใจคำสั่งต่อไป
- คำสั่ง break ทำหน้าที่กระโดดออกจาก loop

ตัวอย่างการใช้งาน for

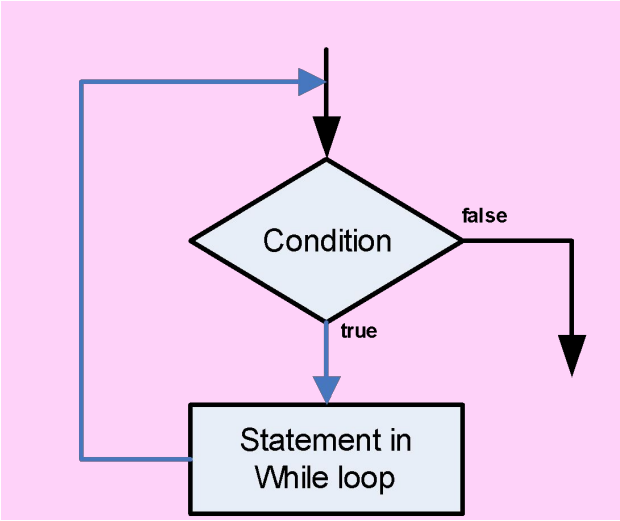
มีลวดหนามยาว 200 เมตร จะล้อมลวดหนามแต่ละด้านอย่างไร ให้มีพื้นที่ที่มากที่สุด และพื้นที่นั้นมีขนาดเท่าไร

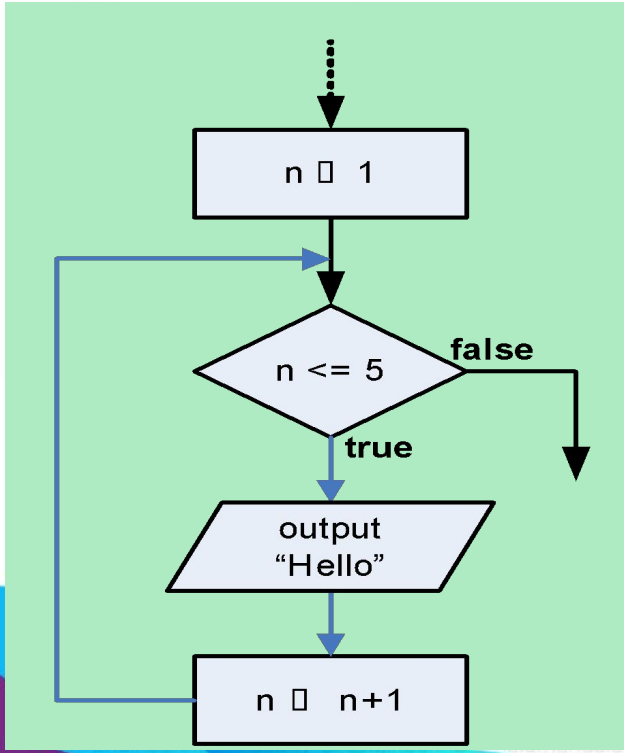
```
1   #include <stdio.h>
2   int main(void)
3   {
4      int x, area, max;
5      printf("Value of x      Area\n");
6      max = 0;
7      for (x=1; x<100; x++)
8      {    area = x*(100-x);
9      printf("%d          %d\n",x,area);
10     if (area > max)   max = area;
11     }
12     printf("Maximum area is %d\n", max);
13     return(0);
14  }
```

ผลลัพธ์การทำงาน

Value of x	Area
1	99
…	…
10	900
11	979
…	…
50	2500
…	…
58	2436
…	…
99	99

Maximum area is 2500





คำสั่ง while

- **โครงสร้างของคำสั่ง while**
  - Condition คือ เงื่อนไขที่ใช้ในการเปรียบเทียบ ถ้าผลลัพธ์ของการเปรียบเทียบเป็นจริง จะมีการเข้าไปทำงานคำสั่งในวงรอบ
  - statement คือ คำสั่งที่จะถูกกระทำเมื่อเงื่อนไขการเปรียบเทียบเป็นจริง
  - คำสั่ง while จะต้องมีการกำหนดค่าเริ่มต้นให้กับ index (ในที่นี้คือตัวแปร n) ก่อนที่จะเข้า loop ดังตัวอย่าง คือ n 1
  - ในวงรอบจะต้องมีคำสั่งสำหรับ update ค่า index เพื่อที่จะทำให้เงื่อนไขมีแนวโน้มจบวงรอบได้ ในที่นี้คือ n n+1

คำสั่ง while

- **ไวยากรณ์ของคำสั่ง while**

  while (condition)

  Statement ;

ตัวอย่างเช่น

```
n = 1;
while (n <= 5)
{
    printf("%d. RMUTT\n", n);
    n++;
}
```

ผลลัพธ์การทำงาน

1. RMUTT
2. RMUTT
3. RMUTT
4. RMUTT
5. RMUTT

- คำสั่ง while สามารถทำซ้ำได้เพียงหนึ่งคำสั่งเท่านั้น ถ้าต้องการให้ทำซ้ำมากกว่าหนึ่งคำสั่ง ต้องใส่คำสั่งเหล่านั้นไว้ภายใต้เครื่องหมาย { }

ตัวอย่างการใช้งาน while

```
1   #include <stdio.h>
2   int main(void)
3   {
4     int  sum, n;
5     sum = 0;
6     n = 1;
7     while (n <= 10)
8     {
9         sum = sum+n;
10        n++;
11    }
12    printf("Summation of 1-10 is : %d", sum);
13    return(0);
14  }
```

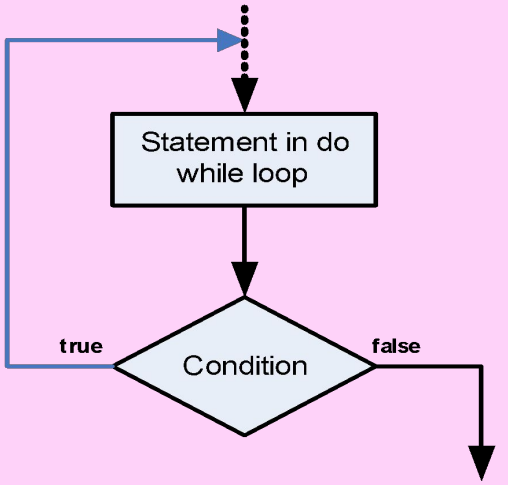
ผลลัพธ์การทำงาน

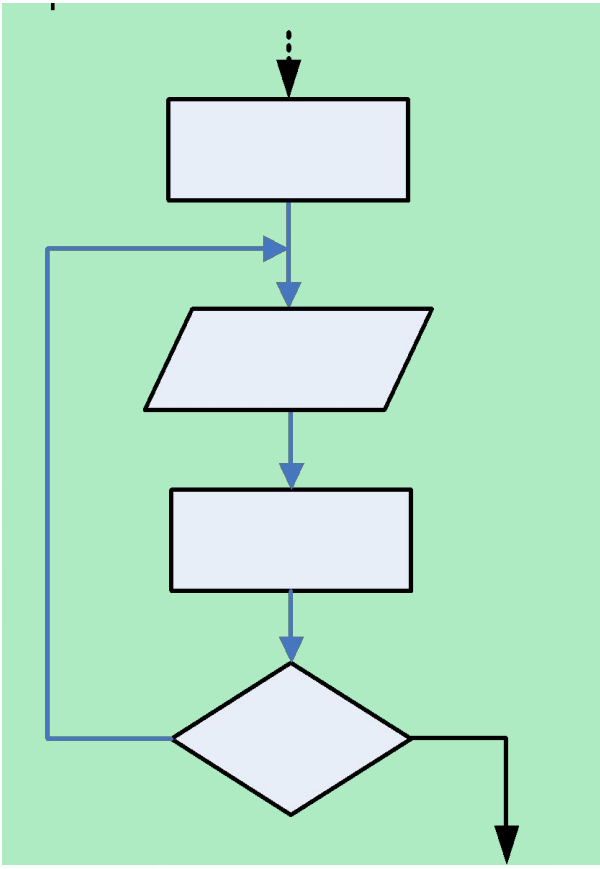
Summation of 1-10 is 55

- คำสั่ง while สามารถทำซ้ำได้เพียงหนึ่งคำสั่งเท่านั้น ถ้าต้องการให้ทำซ้ำมากกว่าหนึ่งคำสั่ง ต้องใส่คำสั่งเหล่านั้นไว้ภายใต้เครื่องหมาย { }

คำสั่ง do-while

- **โครงสร้างของคำสั่ง do-while**
  - Condition คือ เงื่อนไขที่ใช้เปรียบเทียบซึ่งถ้าผลลัพธ์ของการเปรียบเทียบเป็นจริง จะมีการเข้าไปทำงานคำสั่งในวงรอบ ถ้าเป็นเท็จจะออกนอกวงรอบ
  - statement คือ คำสั่งที่จะถูกกระทำเมื่อเงื่อนไขการเปรียบเทียบเป็นจริง
  - จุดเด่นของ do-while คือ คำสั่งทำซ้ำที่จะถูกกระทำก่อนอย่างน้อยหนึ่งครั้ง ก่อนจะออกจากวงรอบ
  - ในการ update และ initial ค่าเริ่มต้นยังคงเป็นหลักการเดียวกันกับคำสั่ง while





ตัวอย่างการใช้งาน do-while

- ไวยากรณ์ของคำสั่ง do-while

```
do {
      statement in do-while
   } while (condition);
```

ตัวอย่างเช่น

```
n = 1;
do {
     printf("%d. RMUTT\n", n);
     n++;
   } while (n <= 5);
```

ผลลัพธ์การทำงาน

1. RMUTT
2. RMUTT
3. RMUTT
4. RMUTT
5. RMUTT

ตัวอย่างการใช้งาน do-while

```
do {
     printf("****Menu****\n");
    printf(" 1. Show odd\n");
     printf(" 2. Show even\n");
    printf(" 3. Exit\n");
     printf(" Please enter number [1-3] :
");
     scanf("%d", &choice);

    ...
   } while (choice != 3);
```

- หมายเหตุ โปรแกรมจะวนไปรับข้อมูลไปเรื่อยๆ จนกว่า ตัวแปร choice จะมีค่าเท่ากับ 3 จึงจะออกจากการวนรอบ

สรุปการควบคุมทิศทางการทำงานของโปรแกรม
การทำซ้ำ

- **หลักการในการทำซ้ำนั้นจะประกอบไปด้วยสิ่งสำคัญ 3 สิ่ง คือ**
  - การกำหนดค่าเริ่มต้นให้กับค่าดัชนี (Index)
  - เงื่อนไข (Condition) ที่จะตัดสินว่า จะให้ทำซ้ำในวงรอบหรือไม่
  - การปรับค่าดัชนี (Update index) ภายในวงรอบ

```
i = 1;
while (i <= 5)
{
  printf("RMUTT");
  i++;
}
```

```
i = 1;
do {
  printf("RMUTT");
  i++;
} while (i<=5);
```

```
for(i=1; i<=5; i++)
{
  printf("RMUTT");
}
```

จบบทที่ 4-2

การควบคุมทิศทางการทำงานของโปรแกรม
การทำซ้ำ (Repetition)