

แผนการสอนประจำบทที่ 4

คำสั่งวนซ้ำ

หัวข้อสำคัญ

1. การวนซ้ำโดยใช้คำสั่ง while
2. การวนซ้ำโดยใช้คำสั่ง do..while
3. การวนซ้ำโดยใช้คำสั่ง for
4. การซ้อนคำสั่งวนซ้ำ (Nested-loop)

วัตถุประสงค์เชิงพฤติกรรมการเรียนการสอน

1. ผู้เรียนสามารถอธิบายความหมายของคำสั่ง while, do..while และ for ได้
2. ผู้เรียนสามารถการทำงานของคำสั่งวนซ้ำได้เป็นอย่างดี
3. ผู้เรียนสร้างคำสั่งวนซ้ำได้อย่างถูกต้อง

วิธีการสอนและกิจกรรมการเรียนการสอน

1. การบรรยาย
2. การทำแบบฝึกหัด

สื่อที่ใช้ประกอบการสอน

1. เอกสารประกอบการสอน
2. เครื่องคอมพิวเตอร์
3. เครื่องฉายภาพนิ่ง

การวัดและประเมินผล

1. สังเกตจากความสนใจของผู้เรียน
2. ประเมินจากการตอบคำถามของผู้เรียนและกิจกรรมในชั้นเรียน
3. การทำแบบฝึกหัดท้ายบท

บทที่ 4

คำสั่งวนซ้ำ

การจัดภาพในผังงานแบ่งเป็น 3 ประเภทใหญ่

1. การทำงานตามลำดับ (Sequence)
2. การเลือก (Selection)
3. การทำงานซ้ำ (Iteration)

จุดประสงค์ของคำสั่งวนซ้ำ

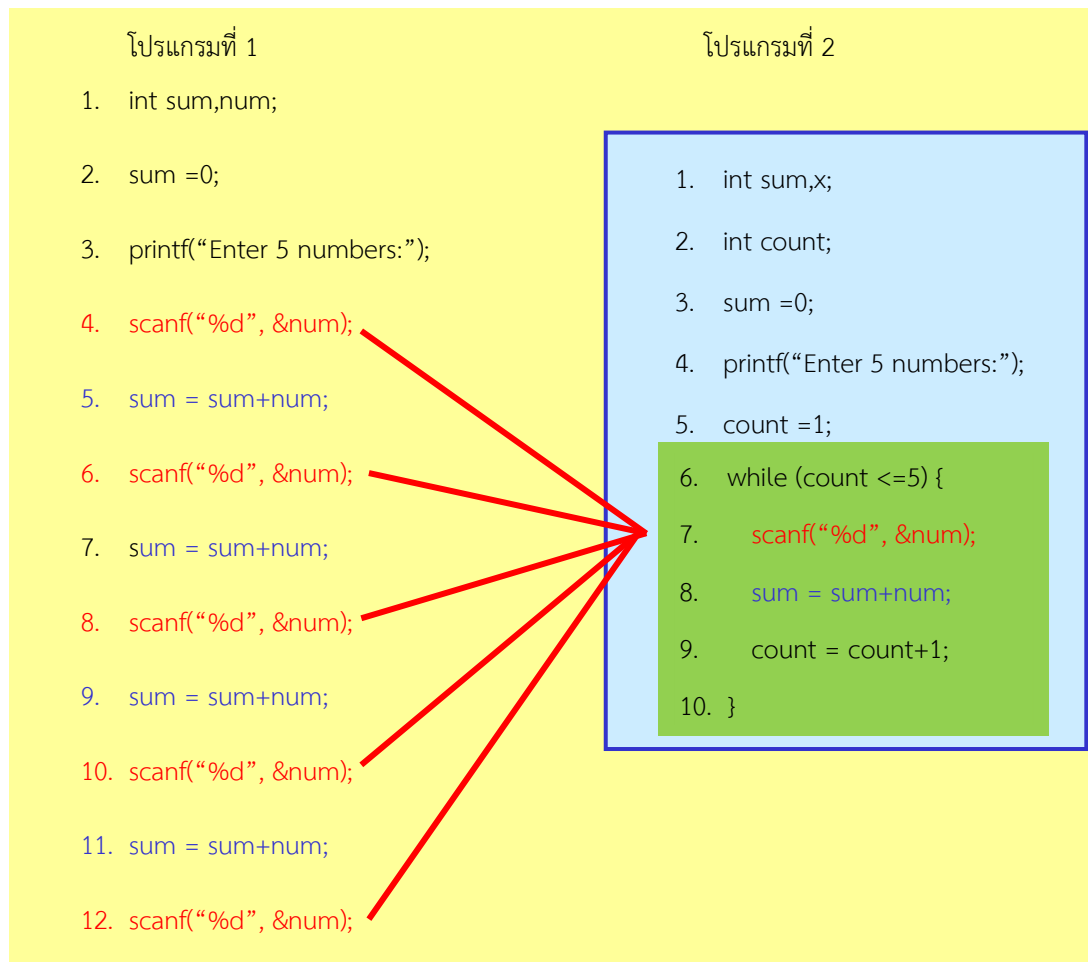
ปัญหา: ต้องการหาผลบวกของจำนวนเลข 5 ตัวที่ป้อนผ่านทางแป้นพิมพ์ ต้องประกาศตัวแปรจำนวน 5 ตัวแปร ตามบรรทัดที่ 2 คำสั่งการหาผลบวกแสดงดัง 5 คำสั่งด้านล่างต่อไปนี้

1. `int sum;`
2. `int num1,num2,num3,num4,num5;`
3. `printf("Enter 5 numbers:");`
4. `scanf("%d%d%d%d%d", &num1,&num2,&num3,&num4,&num5);`
5. `sum = num1+num2+num3+num4+num5;`

จากรูปคำสั่งการหาผลบวก 2 โปรแกรมดังภาพ 4.1 สังเกตว่าโปรแกรมให้ผลลัพธ์เหมือนกัน แต่โปรแกรมที่ 2 มีจำนวนบรรทัดสั้นกว่าและเข้าใจง่ายกว่าเมื่อปรับใช้การเขียนเป็นวนซ้ำ

การทำงานแบบวนซ้ำ หรือ วนลูป (looping structure) แบ่งออกเป็น 2 รูปแบบ โดยพิจารณาจากผลลัพธ์ของเงื่อนไขที่ใช้ในการวนซ้ำ (conditional loop) คือ ทำงานภายในคำสั่งวนซ้ำเมื่อเงื่อนไขจะมีค่าความจริงเป็นจริง (true) และ ทำงานภายในคำสั่งวนซ้ำเมื่อเงื่อนไขจะมีค่าจริงเป็นเท็จ (false)

เปรียบเทียบชุดคำสั่งการหาผลบวก 2 วิธี



ภาพ 4.1 การเปรียบเทียบคำสั่งโปรแกรมการหาผลบวก

4.1 โครงสร้างควบคุมแบบทำซ้ำ

คำสั่งที่ใช้ในโครงสร้างควบคุมแบบทำซ้ำในภาษาซีมีหลายรูปแบบ ดังนี้

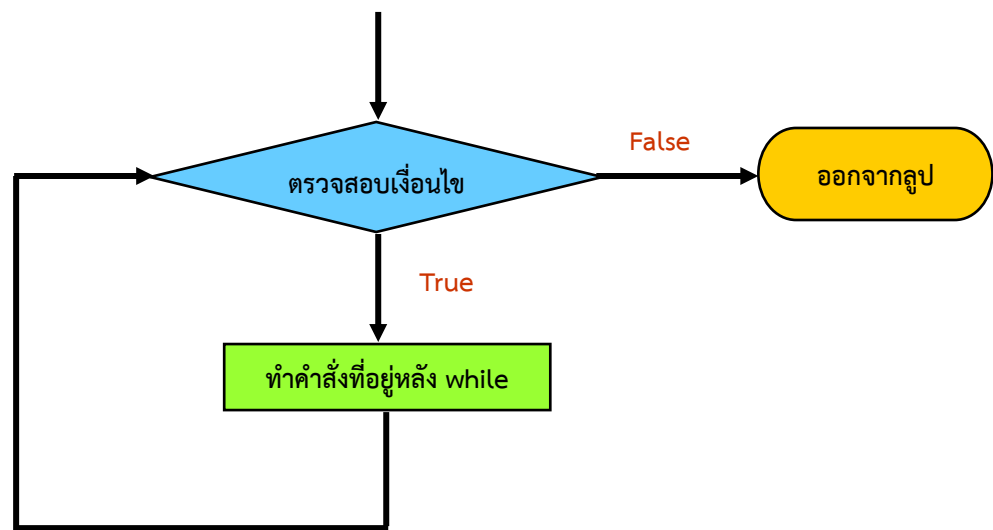
1. คำสั่งวนซ้ำ while
2. คำสั่งวนซ้ำ do..while
3. คำสั่ง for

4.1.1 คำสั่ง while (ใช้เงื่อนไขจริงในการวนซ้ำ)

คำสั่ง while เป็นคำสั่งที่ใช้ในการวนซ้ำ (loop) ชุดคำสั่งที่ต้องการโดยที่กำหนดเงื่อนไขให้ตรวจสอบเพื่อหยุดการวนซ้ำ ใช้ในกรณีที่เรารู้จำนวนรอบในการวนซ้ำที่แน่นอนใช้เงื่อนไขจริงในการวนซ้ำ เงื่อนไขต้องเป็นนิพจน์แบบตรรกะ (boolean expression) ซึ่งจะมีผลลัพธ์ได้ 2 รูปแบบ คือ

- 1) จริง (true) => ประมวลผลชุดคำสั่งที่อยู่ใน while แบบวนซ้ำ
- 2) เท็จ (false) => หยุดการวนซ้ำและประมวลผลคำสั่งถัดไป

แผนผังแสดงขั้นตอนการทำงานของคำสั่ง while แสดงดังภาพ 4.2



ภาพ 4.2 แผนผังการทำงานของคำสั่ง while

รูปแบบคำสั่ง while	ตัวอย่าง
กรณีหลังเงื่อนไขให้ทำงาน 1 คำสั่ง while (เงื่อนไข) คำสั่ง A;	num=1; while (num < 5) printf("number is %d \n",num);
กรณีหลังเงื่อนไขให้ทำงาน > 1 คำสั่ง while (เงื่อนไข) { คำสั่ง A ₁ ; คำสั่ง A ₂ ; คำสั่ง A _n ; }	num=1; while (num < 5) { printf("number is %d \n",num); printf("—While Loop—"); }

ตัวอย่างการสร้างเงื่อนไขของคำสั่ง while

1) ต้องการใช้คำสั่ง while วนซ้ำ เมื่อตัวแปรชนิดจำนวนเต็ม (int) จากตัวอย่างด้านล่าง มีการกำหนดค่าเริ่มต้นของ num เป็น 0 และภายในคำสั่งวนซ้ำมีการเพิ่มค่า num ครั้งละ 1 ดังนั้นค่าตัวแปร num ที่ทำงานในคำสั่งวนซ้ำจะต้องมีค่าน้อยกว่าหรือเท่ากับ 100 คือ num มีค่าเท่ากับ 0-100 และค่า num ค่าแรกที่ไม่ทำภายในคำสั่งวนซ้ำ คือ ค่า num = 101 เพราะเป็นค่าแรกที่เงื่อนไขของ while เป็นเท็จจึงออกจากการวนซ้ำ ดังภาพ 4.3

```
num = 0;
while (num<=100)
{
    คำสั่งที่ 1;
    คำสั่งที่ 2;
    .....
    คำสั่งที่ n;
    num++;
}
```

ภาพ 4.3 ตัวอย่างการสร้างเงื่อนไข while แบบที่ 1

2) ต้องการใช้คำสั่ง while วนซ้ำ เมื่อตัวแปรชนิดตัวอักขระ (char) จากตัวอย่างโปรแกรมจะทำงานวนซ้ำภายในคำสั่ง while เมื่อค่าตัวแปร data ต้องมีค่าไม่เท่ากับ N เท่านั้น และเมื่อมีการรับค่า data เข้ามาเป็นค่า N โปรแกรมจะออกจากการวนซ้ำทันที ดังภาพ 4.4

```
while (data != 'N')
{
    คำสั่งที่ 1;
    คำสั่งที่ 2;
    .....
    คำสั่งที่ n;
}
```

ภาพ 4.4 ตัวอย่างการสร้างเงื่อนไข while แบบที่ 2

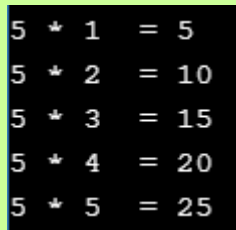
3) ต้องการใช้คำสั่ง while วนซ้ำ เมื่อตัวแปรชนิดตัวสายอักขระหรือข้อความ (string) จากตัวอย่างโปรแกรมจะทำงานวนซ้ำภายในคำสั่ง while เมื่อค่าตัวแปร subject ต้องมีค่าไม่เท่ากับ Dev-C เท่านั้น จะใช้การกำหนดเงื่อนไขเหมือน char ไม่ได้ แต่การเปรียบเทียบข้อมูลชนิด string นั้น จำเป็นต้องมีการเรียกฟังก์ชันชื่อ strcmp ที่อยู่ใน library ชื่อ string.h มาใช้งานด้วย ซึ่งโปรแกรมจะพิจารณาค่าของ subject โดย

ทำงานวนซ้ำภายในคำสั่ง while เมื่อค่าตัวแปร subject ต้องมีค่าไม่เท่ากับ Dev-C เท่านั้น เมื่อมีการรับค่า subject เข้ามาเป็นค่า Dev-C โปรแกรมจะออกจากการวนซ้ำทันที ดังภาพ 4.5

```
while (strcmp(subject,"Dev-C") != 0)
{
    คำสั่งที่ 1;
    คำสั่งที่ 2;
    .....
    คำสั่งที่ n;
}
```

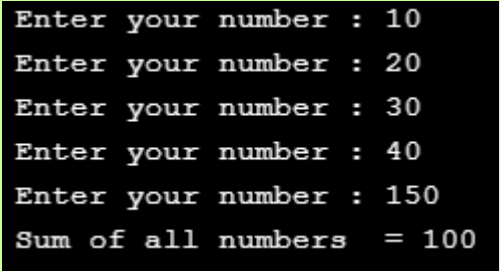
ภาพ 4.5 ตัวอย่างการสร้างเงื่อนไข while แบบที่ 3

ตัวอย่างคำสั่ง while ที่ 1 สูตรคูณแม่ 5

โปรแกรม	ผลลัพธ์
<pre> 1. #include <stdio.h> 2. main() 3. { 4. int num, value; 5. num = 1; 6. while (num <= 5) 7. { 8. value = 5*num; 9. printf ("5 * %d = %d \n",num,value); 10. num++; 11. } 12. }</pre>	

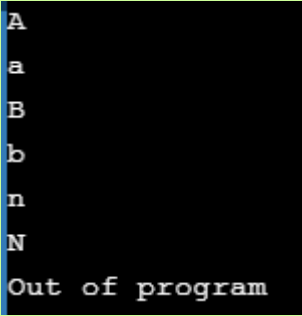
ตัวอย่างคำสั่ง while ที่ 2 มีการรับค่าจำนวนเต็ม

โปรแกรมรับค่าตัวเลขจำนวนเต็มที่มีค่าน้อยกว่าหรือเท่ากับ 100 ไม่จำกัดจำนวนแล้ว
คำนวณหาผลรวมของค่าตัวเลขที่รับเข้ามา

โปรแกรม	ผลลัพธ์
<pre> 1. #include <stdio.h> 2. main() 3. { 4. int num = 0 ,sum = 0; 5. while (num <= 100) 6. { 7. printf("Enter your number : "); 8. scanf("%d",&num); 9. sum = sum + num; 10. } 11. printf("Sum of all numbers = %d", sum-num); 12. }</pre>	 <pre> Enter your number : 10 Enter your number : 20 Enter your number : 30 Enter your number : 40 Enter your number : 150 Sum of all numbers = 100</pre>

ตัวอย่างคำสั่ง while ที่ 3 มีการรับค่าอักขระ

โปรแกรมรับค่าตัวเลขจำนวนเต็มที่มีค่าน้อยกว่าหรือเท่ากับ 100 ไม่จำกัดจำนวนแล้ว
คำนวณหาผลรวมของค่าตัวเลขที่รับเข้ามา

โปรแกรม	ผลลัพธ์
<pre> 1. #include <stdio.h> 2. main() 3. { 4. char data; 5. while (data != 'N') 6. { 7. data = getchar(); 8. } 9. printf("Out of program"); 10. }</pre>	 <pre> A a B b n N Out of program</pre>

4.1.2 คำสั่ง do-while (ใช้เงื่อนไขจริงในการวนซ้ำ)

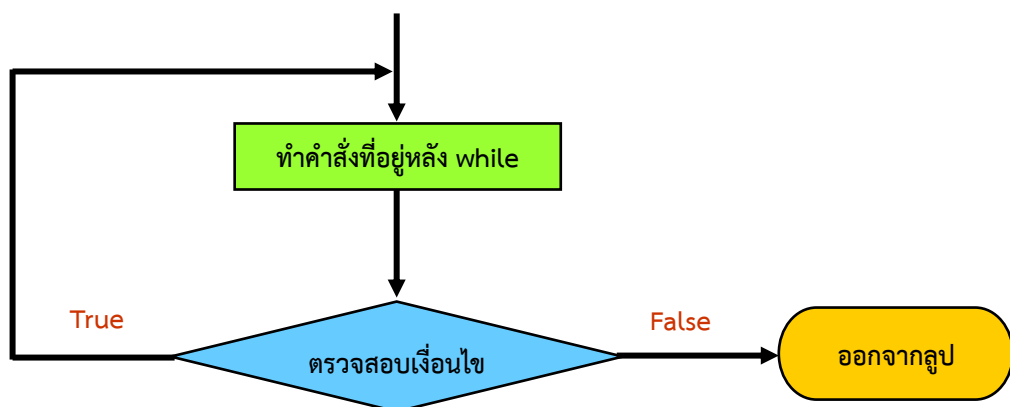
คำสั่ง do..while เป็นคำสั่งที่ใช้ในการวนซ้ำ (loop) ชุดคำสั่งที่ต้องการโดยที่กำหนดเงื่อนไขให้ตรวจสอบเพื่อหยุดการวนซ้ำ ใช้ในกรณีที่เรารู้จำนวนรอบในการวนซ้ำที่แน่นอนใช้ เงื่อนไขเป็นจริงในการวนซ้ำ เงื่อนไขต้องเป็น นิพจน์แบบตรรกะ ซึ่งจะมีผลลัพธ์ได้ 2 รูปแบบ เช่นเดียวกับคำสั่ง while คือ

- 1) จริง (true) => ประมวลผลชุดคำสั่งที่อยู่ใน do...while
- 2) เท็จ (false) => หยุดการวนซ้ำและประมวลผลคำสั่งถัดไป

แต่ข้อแตกต่างระหว่างคำสั่ง do.. while และคำสั่ง while คือ หลักการทำงานของการทำงานโดยใช้คำสั่ง do-while คือ โปรแกรมจะทำคำสั่งที่ 1 ก่อน 1 รอบ แล้วจึงทำการตรวจสอบเงื่อนไขว่าเป็นจริงหรือไม่ หากเงื่อนไขเป็นจริง (true หรือ 1) จะทำคำสั่งวนซ้ำภายใน do..while แต่เมื่อผลการตรวจสอบเงื่อนไขเป็นเท็จ (false หรือ 0) จะออกทำการวนซ้ำทันที โดยแผนผังแสดงขั้นตอนการทำงานของคำสั่ง while แสดงดังภาพ 4.6

หมายเหตุ

- 1) สิ่งทีคำสั่ง do.. while แตกต่างจากคำสั่ง while คือ จำนวนรอบในการทำงานใน Loop (น้อยที่สุด)
- 2) คำสั่ง while เข้าไปทำงานในคำสั่งวนซ้ำอย่างน้อย 0 รอบ ก่อนออกจาก loop เนื่องจากเงื่อนไขเป็น เท็จ ไม่สามารถเข้า loop ได้ (while)
- 3) คำสั่ง do.. while เข้าไปทำงานในคำสั่งวนซ้ำอย่างน้อย 1 รอบ ก่อนออกจาก Loop แม้เงื่อนไขเป็น เท็จ แต่ตรวจสอบเงื่อนไขหลังเข้า loop แล้ว (do...while)



ภาพ 4.6 แผนผังการทำงานของคำสั่ง do..while

รูปแบบคำสั่ง do..while	ตัวอย่าง
กรณีหลังเงื่อนไขให้ทำงาน 1 คำสั่ง do คำสั่ง A; while (เงื่อนไข);	num=1; do printf("number is %d \n",num); while (num < 5);
กรณีหลังเงื่อนไขให้ทำงาน > 1 คำสั่ง do { คำสั่ง A ₁ ; คำสั่ง A ₂ ; คำสั่ง A _n ; } while (เงื่อนไข);	num=1 do { printf("number is %d \n",num); printf("—Do While Loop—"); } while (num < 5);

ตัวอย่างการสร้างเงื่อนไขของคำสั่ง do..while

1) ต้องการใช้คำสั่ง do..while วนซ้ำ เมื่อตัวแปรชนิดจำนวนเต็ม (int) จากตัวอย่างด้านล่าง มีการกำหนดค่าเริ่มต้นของ num เป็น 0 และภายในคำสั่งวนซ้ำมีการเพิ่มค่า num ครั้งละ 1 ดังนั้นค่าตัวแปร num ที่ทำงานในคำสั่งวนซ้ำจะต้องมีค่าน้อยกว่าหรือเท่ากับ 100 คือ num มีค่าเท่ากับ 0-100 และค่า num ค่าแรกที่ไม่ทำภายในคำสั่งวนซ้ำ คือ ค่า num = 101 เพราะเป็นค่าแรกที่เงื่อนไขของ while เป็นเท็จจึงออกจากการวนซ้ำ ดังภาพ 4.7

```

num = 0;
do {
    คำสั่งที่ 1;
    คำสั่งที่ 2;
    .....
    คำสั่งที่ n;
    num++;
} while (num<=100);

```

ภาพ 4.7 ตัวอย่างการสร้างเงื่อนไข do...while แบบที่ 1

2) ต้องการให้คำสั่ง while วนซ้ำ เมื่อตัวแปรชนิดตัวอักษร (char) จากตัวอย่างโปรแกรมจะทำงานวนซ้ำภายในคำสั่ง while เมื่อค่าตัวแปร data ต้องมีค่าไม่เท่ากับ N เท่านั้น และเมื่อมีการรับค่า data เข้ามาเป็นค่า N โปรแกรมจะออกจากการวนซ้ำทันที ดังภาพ 4.8

```
do {  
    คำสั่งที่ 1;  
    คำสั่งที่ 2;  
    .....  
    คำสั่งที่ n;  
} while (data != 'N');
```

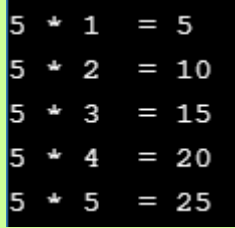
ภาพ 4.8 ตัวอย่างการสร้างเงื่อนไข do...while แบบที่ 2

3) ต้องการให้คำสั่ง while วนซ้ำ เมื่อตัวแปรชนิดตัวสายอักขระหรือข้อความ (string) จากตัวอย่างโปรแกรมจะทำงานวนซ้ำภายในคำสั่ง while เมื่อค่าตัวแปร subject ต้องมีค่าไม่เท่ากับ Dev-C เท่านั้น จะใช้การกำหนดเงื่อนไขเหมือน char ไม่ได้ แต่การเปรียบเทียบข้อมูลชนิด string นั้น จำเป็นต้องมีการเรียกฟังก์ชันชื่อ strcmp ที่อยู่ใน library ชื่อ string.h มาใช้งานด้วย ซึ่งโปรแกรมจะพิจารณาค่าของ subject โดยทำงานวนซ้ำภายในคำสั่ง while เมื่อค่าตัวแปร subject ต้องมีค่าไม่เท่ากับ Dev-C เท่านั้น เมื่อมีการรับค่า subject เข้ามาเป็นค่า Dev-C โปรแกรมจะออกจากการวนซ้ำทันที ดังภาพ 4.9

```
do {  
    คำสั่งที่ 1;  
    คำสั่งที่ 2;  
    .....  
    คำสั่งที่ n;  
} while (strcmp(subject, "Dev-C") != 0);
```

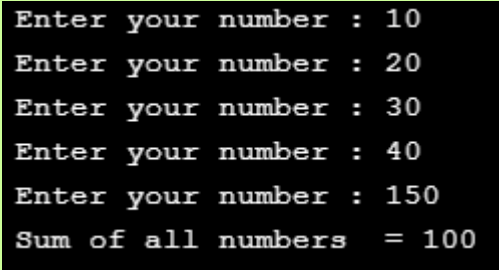
ภาพ 4.9 ตัวอย่างการสร้างเงื่อนไข do...while แบบที่ 3

ตัวอย่างคำสั่ง while ที่ 1 สูตรคูณแม่ 5

โปรแกรม	ผลลัพธ์
<pre>#include <stdio.h> main() { int num, value; num = 1; do { value = 5*num; printf("5 * %d = %d \n",num,value); num++; } while (num <= 5); }</pre>	 <pre>5 * 1 = 5 5 * 2 = 10 5 * 3 = 15 5 * 4 = 20 5 * 5 = 25</pre>

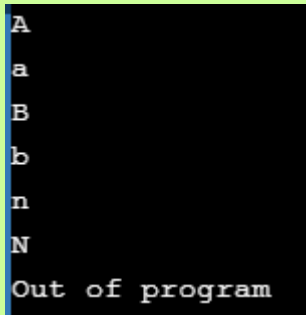
ตัวอย่างคำสั่ง do..while ที่ 2 มีการรับค่าจำนวนเต็ม

โปรแกรมรับค่าตัวเลขจำนวนเต็มที่มีค่าน้อยกว่าหรือเท่ากับ 100 ไม่จำกัดจำนวนแล้ว
คำนวณหาผลรวมของค่าตัวเลขที่รับเข้ามา

โปรแกรม	ผลลัพธ์
<pre>1. #include <stdio.h> 2. main() 3. { 4. int num = 0 ,sum = 0; 5. do { 6. printf("Enter your number : "); 7. scanf("%d",&num); 8. sum = sum + num; 9. } while (num <= 100); 10. printf("Sum of all numbers = %d", sum-num); 11. }</pre>	 <pre>Enter your number : 10 Enter your number : 20 Enter your number : 30 Enter your number : 40 Enter your number : 150 Sum of all numbers = 100</pre>

ตัวอย่างคำสั่ง do..while ที่ 3 มีการรับค่าอักขระ

โปรแกรมรับค่าตัวเลขจำนวนเต็มที่มีค่าน้อยกว่าหรือเท่ากับ 100 ไม่จำกัดจำนวนแล้ว
คำนวณหาผลรวมของค่าตัวเลขที่รับเข้ามา

โปรแกรม	ผลลัพธ์
<pre> 1. #include <stdio.h> 2. main() 3. { 4. char data; 5. do 6. { 7. data = getchar(); 8. } while (data != 'N'); 9. printf("Out of program"); 10. }</pre>	

4.1.3 คำสั่ง for

คำสั่ง for เป็นคำสั่งที่ใช้ในการวนซ้ำ (วนลูป) ชุดคำสั่งที่ต้องการโดยที่ระบุจำนวนรอบที่แน่นอนผ่านตัวแปรประเภทจำนวนเต็มหรือตัวอักษร (character) เงื่อนไขต้องเป็น นิพจน์แบบตรรกะ (boolean expression) ซึ่งจะมีผลลัพธ์ได้ 2 รูปแบบ คือ

- 1) จริง (true) => ประมวลผลชุดคำสั่งที่อยู่ใน for
- 2) เท็จ (false) => หยุดการวนซ้ำและประมวลผลคำสั่งถัดไป

สามารถแบ่งการวนซ้ำออกเป็น 2 รูปแบบ คือ


- 1) แบบเพิ่มค่าในการวน loop => เช่น i++, i+=2
- 2) แบบลดค่าในการวน loop => เช่น i--, i/=2

รูปแบบคำสั่ง for	ตัวอย่าง
กรณีหลังเงื่อนไขให้ทำงาน 1 คำสั่ง for (ค่าเริ่มต้น; ค่าสิ้นสุด; การควบคุมค่า) คำสั่ง A;	for (count = 1; count<=5; count++) printf(“%d”,count);
กรณีหลังเงื่อนไขให้ทำงาน > 1 คำสั่ง while (ค่าเริ่มต้น; ค่าสิ้นสุด; การควบคุมค่า){ คำสั่ง A ₁ ; คำสั่ง A ₂ ; คำสั่ง A _n ; }	for (num= 1; num<=5; num++) { printf(“Enter your number %d :”,num); printf(“---For Loop---”); }


■ ค่าเริ่มต้น (initial value) และ ค่าสิ้นสุด (final value) เป็นข้อมูลเริ่มต้นและสิ้นสุดที่ใช้ในการวน Loop ทั้ง 2 ค่า ต้องเป็นข้อมูลที่มีชนิดเดียวกัน ค่าคงที่, ตัวแปร หรือนิพจน์ที่ให้ค่าเป็นตัวเลขแบบจำนวนเต็มก็ได้

■ การควบคุมค่า (control value) เป็นตัวแปรนับจำนวนรอบของวนลูป จะต้องมีชนิดเป็นตัวเลขจำนวนเต็มหรือตัวอักษร (character) มีทั้งการเพิ่มและลดค่า

ตัวอย่างคำสั่ง for ที่ 1

โปรแกรม	ผลลัพธ์
<pre> 1. #include <stdio.h> 2. main() 3. { 4. int count; 5. for (count = 1; count<=5; count++) 6. printf(“%d”,count); 7. }</pre>	

ตัวอย่างคำสั่ง for ที่ 2

โปรแกรม	ผลลัพธ์
<pre> 1. #include <stdio.h> 2. main() 3. { 4. char ch; 5. for (ch='Z';ch>='A'; ch--) 6. printf(“%c”,ch); 7. }</pre>	

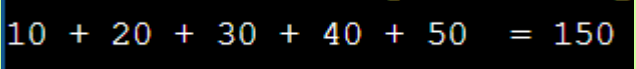
ตัวอย่างคำสั่ง for ที่ 3

โปรแกรม	ผลลัพธ์
<pre> 1. #include <stdio.h> 2. main() 3. { 4. int num,i; 5. printf("Please input number : "); 6. scanf("%d",&num); 7. for (i=num; i>=0; i-=2) 8. printf("number = %d\n",i); 9. }</pre>	<pre> Please input number : 10 number = 10 number = 8 number = 6 number = 4 number = 2 number = 0</pre>

ตัวอย่างคำสั่ง for ที่ 4

โปรแกรม	ผลลัพธ์
<pre> 1. #include <stdio.h> 2. main() 3. { 4. int num,value,sum=0; 5. for (num= 1; num<=5; num++) 6. { 7. printf("Enter your number %d :",num); 8. scanf("%d", &value); 9. sum = sum + value; 10. } 11. printf("Sum of all numbers : %d",sum); 12. }</pre>	<pre> Enter your number 1 :10 Enter your number 2 :20 Enter your number 3 :30 Enter your number 4 :40 Enter your number 5 :50 Sum of all numbers : 150</pre>

ตัวอย่างคำสั่ง for ที่ 5

โปรแกรม	ผลลัพธ์
<pre> 1. #include <stdio.h> 2. main() 3. { 4. int num,sum=0; 5. for (num=10; num<=50; num +=10) 6. { 7. if(num==50) printf("%d ",num); 8. else printf("%d + ",num); 9. sum = sum + num; 10. } 11. printf(" = %d",sum); 12. }</pre>	

การเปรียบเทียบคำสั่ง for กับ while และ do..while

คำสั่ง for	คำสั่ง while	คำสั่ง do..while
<pre> 1. for (i=1; i<=10; i++) 2. { 3. num := i * 2; 4. printf("%d ",num, ', '); 5. }</pre>	<pre> 1. i=1; 2. while (i<=10) 3. { 4. num := i * 2; 5. printf("%d ",num, ', '); 6. i++; 7. }</pre>	<pre> 1. i=1; 2. do 3. { 4. num := i * 2; 5. printf("%d ",num, ', '); 6. i++; 7. } while (i<=10);</pre>

4.1.4 คำสั่ง break และ continue

1) คำสั่ง break

เป็นคำสั่งที่สั่งให้ออกจากคำสั่งวนซ้ำและคำสั่ง switch.. case โดยทันที

รูปแบบคำสั่ง

```
break;
```

ตัวอย่างคำสั่ง break

โปรแกรม	ผลลัพธ์
<pre> 1. #include<stdio.h> 2. main() 3. { 4. int i=0; 5. while(i<=10) { 6. printf("i = %d\t",i); 7. if (i==5) break; /* break if i=5 */ 8. else printf("%d not using break\n",i); 9. i++; 10. } 11. printf("\n...End of loop while..."); 12. }</pre>	<pre> i = 0 0 not using break i = 1 1 not using break i = 2 2 not using break i = 3 3 not using break i = 4 4 not using break i = 5 ...End of loop while...</pre>

2) continue

เป็นคำสั่งที่สั่งให้คำสั่งวนซ้ำจบรอบการทำงานและกลับไปทำงานที่คำสั่งแรกของลูปโดยทันที

รูปแบบคำสั่ง

```
continue;
```

ตัวอย่างคำสั่ง continue

โปรแกรม	ผลลัพธ์
<pre> 1. #include<stdio.h> 2. main() 3. { 4. int i=0; 5. while(i<=10) { 6. i++; 7. printf("i = %d\t",i); 8. if (i==5) continue; /* out of loop if i=5 */ 9. else printf("%d not using continue\n",i); 10. } 11. printf("\n...End of loop while..."); 12. }</pre>	<pre> i = 1 1 not using continue i = 2 2 not using continue i = 3 3 not using continue i = 4 4 not using continue i = 5 i = 6 6 not using continue i = 7 7 not using continue i = 8 8 not using continue i = 9 9 not using continue i = 10 10 not using continue i = 11 11 not using continue ...End of loop while...</pre>

4.1.5. คำสั่งซ้อนการวนซ้ำ (Nested-loop)

1) รูปแบบของ Nested for

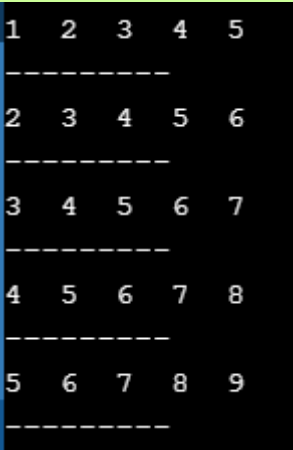
เป็นการวางคำสั่ง for หลายวงซ้อนกัน ลักษณะการทำงาน

เริ่มเข้า loop -> จะทำจาก loop วงนอกเข้าไปหา loop วงในทำงาน

ออกจาก loop -> ทำงาน loop วงในให้เสร็จก่อนจึงจะออกมาทำงานที่ loop วงนอก

```
for (คำสั่งที่ 1.1; คำสั่งที่ 1.2; การควบคุมค่าที่ 1)
{
    คำสั่งที่ 1.1;
    for (คำสั่งที่ 2.1; คำสั่งที่ 2.2; การควบคุมค่าที่ 2)
    {
        คำสั่งที่ 2.1;
        คำสั่งที่ 2.2;
        .....
        คำสั่งที่ 2.n;
    }
    คำสั่งที่ 1.2;
    .....
    คำสั่งที่ 1.n;
}
```

ตัวอย่างคำสั่ง Nested for

โปรแกรม	ผลลัพธ์
<pre> 1. #include<stdio.h> 2. main() 3. { 4. int i,j; 5. for (i=1;i<=5;i++) 6. { 7. for (j=i; j<=i + 4;j++) printf("%d ",j); 8. printf("\n-----\n"); 9. } 10. }</pre>	 <pre> 1 2 3 4 5 ----- 2 3 4 5 6 ----- 3 4 5 6 7 ----- 4 5 6 7 8 ----- 5 6 7 8 9 -----</pre>

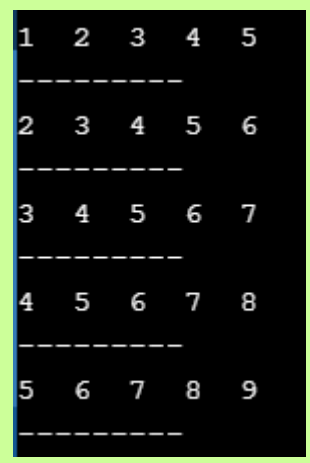
2) รูปแบบของ Nested while

```

while (เงื่อนไขที่ 1)
{
    คำสั่งที่ 1.1;
    while (เงื่อนไขที่ 2)
    {
        คำสั่งที่ 2.1;
        คำสั่งที่ 2.2;
        .....
        คำสั่งที่ 2.n;
    }
    คำสั่งที่ 1.2;
    .....
    คำสั่งที่ 1.n;
}

```

ตัวอย่างคำสั่ง Nested while

โปรแกรม	ผลลัพธ์
<pre> 1. include<stdio.h> 2. main() 3. { 4. int i=1,j; 5. while(i<=5) { 6. j=i; 7. while(j<=i + 4){ 8. printf("%d ",j); 9. j++; 10. } 11. i++; 12. printf("\n-----\n"); 13. } 14. }</pre>	 <pre> 1 2 3 4 5 ----- 2 3 4 5 6 ----- 3 4 5 6 7 ----- 4 5 6 7 8 ----- 5 6 7 8 9 ----- </pre>

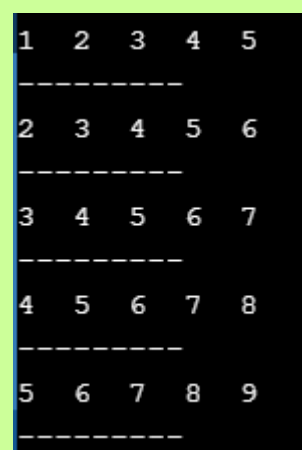
3) รูปแบบของ Nested do...while

```

do
{
    คำสั่งที่ 1.1;
    do
    {
        คำสั่งที่ 2.1;
        คำสั่งที่ 2.2;
        .....
        คำสั่งที่ 2.n;
    } while (เงื่อนไขที่ 2)
    คำสั่งที่ 1.2;
    .....
    คำสั่งที่ 1.n;
} while (เงื่อนไขที่ 1)

```

ตัวอย่างคำสั่ง Nested do...while

โปรแกรม	ผลลัพธ์
<pre> 1. #include<stdio.h> 2. main() 3. { 4. int i=1,j; 5. do{ 6. j=i; 7. do{ 8. printf("%d ",j); 9. j++; 10. } while (j<=i + 4); 11. i++; 12. printf("\n-----\n"); 13. } while(i<=5) ; 14. } </pre>	 <pre> 1 2 3 4 5 ----- 2 3 4 5 6 ----- 3 4 5 6 7 ----- 4 5 6 7 8 ----- 5 6 7 8 9 ----- </pre>

คำถามท้ายบทที่ 4

1. จงอธิบายหลักการทำงานของคำสั่ง while, do..while และ for
2. จงอธิบายความหมายของคำสั่ง break และ continue ในชุดคำสั่งวนซ้ำ
3. จงอธิบายความแตกต่างของคำสั่ง break ภายในคำสั่ง switch และ while
4. จงอธิบายความหมายของค่าเริ่มต้น (initial value) และ ค่าสิ้นสุด (final value) และการควบคุมค่า (control value) ของคำสั่ง For
5. จงประมวลผลโปรแกรมต่อไปนี้พร้อมทั้งแสดงผลลัพธ์ของโปรแกรมที่ปรากฏบนหน้าจอคอมพิวเตอร์ และแสดงตำแหน่งสุดท้ายของ Cursor และกำหนดให้
 - แสดงช่องว่างโดยใช้เครื่องหมาย □
 - แสดงตำแหน่งสุดท้ายของ cursor โดยใช้เครื่องหมาย _

โปรแกรม1	ผลลัพธ์ (แสดงผลลัพธ์ตามบรรทัดบนหน้าจอคอมพิวเตอร์)
<pre> 1. #include<stdio.h> 2. main() 3. { 4. int i=1,sum=0; 5. while (i < 10) { 6. printf("sum = %d \n",sum); 7. if(i<=5) sum = sum + i; 8. else if (i<=7) sum = sum++; 9. else break; 10. i++; 11. } 12. printf("sum out of loop = %d \n",sum); 13. printf("i out of loop = %d \n",i); 14. }</pre>	<pre> 1. 2. 3. 4. 5. 6. 7. 8. 9. 10.</pre>

6. จงประมวลผลโปรแกรมต่อไปนี้พร้อมทั้งแสดงผลลัพธ์ของโปรแกรมที่ปรากฏบนหน้าจอคอมพิวเตอร์ และแสดงตำแหน่งสุดท้ายของ Cursor และกำหนดให้

- แสดงช่องว่างโดยใช้เครื่องหมาย □
- แสดงตำแหน่งสุดท้ายของ cursor โดยใช้เครื่องหมาย _

โปรแกรม1	ผลลัพธ์ (แสดงผลลัพธ์ตามบรรทัดบนหน้าจอคอมพิวเตอร์)
1. #include <stdio.h>	
2. main()	
3. {	1.
4. int i=1, sum=0;	2.
5. do	3.
6. {	4.
7. sum = sum+i; i++;	5.
8. switch(sum%3){	6.
9. case 0 : printf("sum mod 3 = 0\n"); break;	7.
10. case 1 : printf("sum mod 3 = 1\n"); break;	8.
11. default : continue;	9.
12. }	10.
13. } while (i<=9);	11.
14. printf("Sum = %d", sum);	12.
15. }	

7. จงประมวลผลโปรแกรมต่อไปนี้พร้อมทั้งแสดงผลลัพธ์ของโปรแกรมที่ปรากฏบนหน้าจอคอมพิวเตอร์ และแสดงตำแหน่งสุดท้ายของ Cursor และกำหนดให้
- แสดงช่องว่างโดยใช้เครื่องหมาย □
 - แสดงตำแหน่งสุดท้ายของ cursor โดยใช้เครื่องหมาย _

โปรแกรม1	ผลลัพธ์ (แสดงผลลัพธ์ตามบรรทัดบนหน้า จอคอมพิวเตอร์)
<pre> 1. #include<stdio.h> 2. int i,j; 3. main() 4. { 5. 6. i=1; 7. do{ 8. for (j=i;j<=10;j++) 9. printf("%d",j); 10. printf("\n"); 11. i+=2; 12. }while(i<=10); 13. }</pre>	<pre> 1. 2. 3. 4. 5. 6. 7.</pre>

8. จงเขียนโปรแกรมแสดงจำนวนเต็มที่มีคุณสมบัติเป็นเลขคี่จำนวน 10 ค่า โดยเริ่มต้นแสดงค่า 1 เป็นค่าแรก โดยตัวอย่างผลลัพธ์แสดงดังตัวอย่างด้านล่าง

```

1
3
5
7
9
11
13
15
17
19
```

9. จงเขียนโปรแกรมสร้างเมตริกซ์เอกลักษณ์การคูณ (Identity Matrix) โดยกำหนดให้ใช้คำสั่ง FOR ในการเขียนและรับจำนวนแถวของ Identity Matrix จากคีย์บอร์ด โดยตัวอย่างผลลัพธ์แสดงดังตัวอย่างด้านล่าง

```
Please input the number of row : 5
0      1      1      1      1
1      0      1      1      1
1      1      0      1      1
1      1      1      0      1
1      1      1      1      0
_
```

10. จงโปรแกรมให้แสดงผลดังตัวอย่างที่กำหนดให้ โดยตัวอย่างผลลัพธ์แสดงดังตัวอย่างด้านล่าง

```
55555
4444
333
22
1
_
```