คำชี้แจงสำหรับนักเรียน

เอกสารประกอบการเรียน การเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี กลุ่มสาระการเรียนรู้วิทยาศาสตร์และเทคโนโลยี สำหรับนักเรียนชั้นมัธยมศึกษาปีที่ 6 เล่มที่ 5 เรื่อง คำสั่งควบคุมแบบวนซ้ำ เป็นสื่อใช้ประกอบการจัดกิจกรรมการเรียนรู้ นักเรียนควรปฏิบัติ ดังนี้

บทบาทของนักเรียน

1. เอกสารประกอบการเรียนเล่มนี้ สร้างขึ้นเพื่อให้นักเรียนได้ศึกษาด้วยตนเองและใช้ฝึกทักษะการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี นักเรียนต้องมีวินัย มีความรับผิดชอบและมีความซื่อสัตย์ต่อตนเอง
2. นักเรียนต้องศึกษาจุดประสงค์การเรียนรู้ก่อน เพื่อให้ทราบว่าหลังจากศึกษาเนื้อหาในหน่วยการเรียนรู้นี้แล้ว นักเรียนจะต้องมีความสามารถทำอะไรได้บ้าง
3. นักเรียนทำแบบทดสอบก่อนเรียน เพื่อทดสอบความรู้พื้นฐานของตนเอง
4. นักเรียนศึกษาเนื้อหาไปตามลำดับและปฏิบัติกิจกรรมตามคำแนะนำทีละขั้นตอน พร้อมฝึกเขียนโปรแกรมตามตัวอย่าง
5. เมื่อศึกษาเนื้อหาเสร็จสิ้นแล้ว ให้นักเรียนทำแบบทดสอบหลังเรียน
6. หากมีข้อสงสัยใดๆ ให้ปรึกษาครูผู้สอน

หน่วยการเรียนรู้ที่ 5

เรื่อง คำสั่งควบคุมแบบวนซ้ำ **จำนวน 3 ชั่วโมง**

หัวข้อเรื่อง

1. คำสั่ง while
2. คำสั่ง do while
3. คำสั่ง for
4. คำสั่งประกอบการควบคุมทิศทางแบบต่าง ๆ

สาระสำคัญ

1. ภาษาซีมีชุดคำสั่งควบคุมแบบวนซ้ำอยู่ 3 ประเภท ได้แก่ คำสั่ง while, do while และ for โดยการควบคุมแบบวนซ้ำด้วยคำสั่ง while จะทำการตรวจสอบเงื่อนไขก่อนดำเนินการเสมอ ดังนั้น ชุดคำสั่งภายในลูปอาจมิได้ถูกประมวลผลเลยก็ได้ หากตรวจสอบเงื่อนไขครั้งแรกแล้วมีค่าเป็นเท็จ

2. การควบคุมแบบวนซ้ำด้วยคำสั่ง do while ชุดคำสั่งภายในลูป อย่างน้อยจะต้องถูกทำงาน 1 รอบเสมอ ถึงแม้ว่าการตรวจสอบเงื่อนไขในรอบแรกจะเป็นเท็จก็ตาม เนื่องจากลูป do while จะประมวลผลชุดคำสั่งภายในลูปรอบแรกก่อนเสมอ แล้วจึงค่อยตรวจสอบเงื่อนไขถ้าเป็นเท็จก็จะหลุดออกจากลูป

3. การควบคุมแบบวนซ้ำด้วยคำสั่ง for เป็นหนึ่งในชุดคำสั่งควบคุมแบบวนซ้ำของภาษาซี เหมาะกับกรณีที่การวนซ้ำมีจำนวนรอบการทำงานที่แน่นอน

4. คำสั่งประกอบการควบคุมทิศทางการทำงาน เป็นคำสั่งที่ถูกนำไปใช้ร่วมกับคำสั่งอื่น ๆ เช่น นำไปใช้เพื่อหยุดการเลือกทำ หรือออกจากการทำงานของระบบ คำสั่งเหล่านี้ ได้แก่ คำสั่ง break สามารถนำมาใช้เพื่อสั่งให้หลุดออกจากลูปตามเงื่อนไขที่ได้กำหนดไว้ คำสั่ง continue นำไปใช้เพื่อสั่งให้วกกลับไปทำงานซ้ำที่ต้นลูปและ คำสั่ง exit() จะใช้สำหรับออกจากการทำงานของโปรแกรม

จุดประสงค์การเรียนรู้

หลังจากศึกษาเนื้อหาในหน่วยการเรียนรู้นี้แล้วนักเรียนสามารถ

1. บอกรูปแบบและลักษณะการทำงานของคำสั่ง while ได้
2. บอกรูปแบบและลักษณะการทำงานของคำสั่ง do while ได้
3. บอกรูปแบบและลักษณะการทำงานของคำสั่ง for ได้
4. เลือกใช้งานคำสั่งควบคุมแบบวนซ้ำได้เหมาะสม
5. นำคำสั่งประกอบการควบคุมทิศทางแบบต่างๆ ไปใช้งานได้ถูกต้อง
6. เขียนโปรแกรมด้วยภาษาซี โดยใช้คำสั่งควบคุมแบบวนซ้ำได้

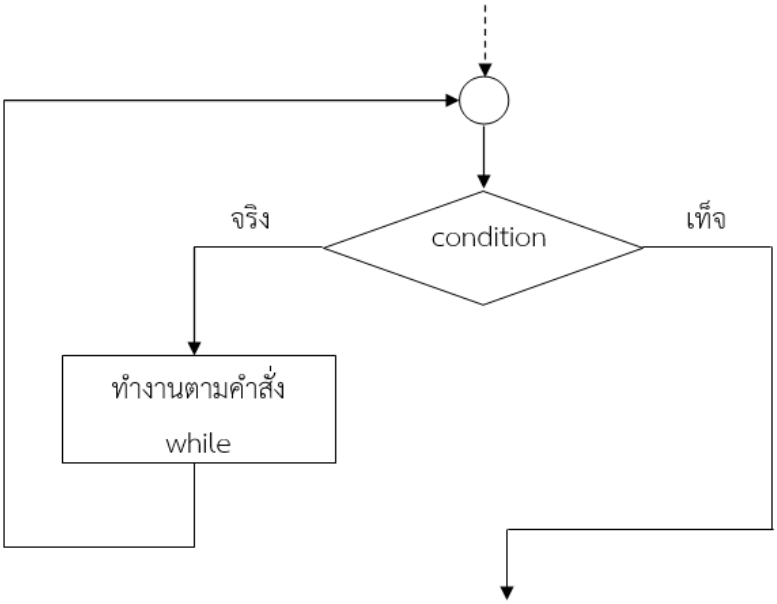
หัวเรื่องที่ 5.1 คำสั่ง while

- เวลาเรียน 30 นาที -

จากโปรแกรมที่ผ่านมา ล้วนเป็นการประมวลผลชุดคำสั่งเพียงรอบเดียวทั้งสิ้น แต่ในความเป็นจริงแล้ว เราสามารถสั่งให้คอมพิวเตอร์ประมวลผลชุดคำสั่งซ้ำ ๆ ได้ เรียกว่า กระบวนการทำซ้ำ หรือ การทำงานแบบวนลูป (Loop) เช่น สร้างลูปของรายการเมนูเพื่อให้ผู้ใช้สามารถใช้งานโปรแกรมไปได้เรื่อย ๆ จนกว่าจะเลือกรายการจบการทำงาน ดังนั้นจะพบว่าจำนวนรอบการทำงานของลูปที่สร้างขึ้นนั้น ขึ้นอยู่กับเงื่อนไขที่กำหนดขึ้น คำสั่งในภาษาซีที่ใช้สำหรับควบคุมทิศทางแบบวนซ้ำ ได้แก่ while, do-while และ for ซึ่งแสดงรายละเอียดในหัวข้อต่อไปนี้

1. ลักษณะการทำงานของคำสั่ง while

คำสั่งวนลูปแบบ while จะเริ่มต้นทำงานจากการตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริง จึงจะทำงานตามคำสั่งของ while เมื่อทำงานเสร็จแล้วก็จะวนกลับไปตรวจสอบเงื่อนไขใหม่ เป็นเช่นนี้ไปเรื่อย ๆ จนกว่าเงื่อนไขจะเป็นเท็จจึงจะหลุดออกจากการทำงานของลูป while สามารถสรุปการทำงานของลูป while ตามผังงาน (Flowchart) ดังรูปต่อไปนี้



นิพจน์ที่นำมาใช้ตรวจสอบเงื่อนไข สามารถใช้ตัวดำเนินการเปรียบเทียบและตัวดำเนินการทางตรรกะได้ และจากคุณลักษณะการทำงานเป็นรอบด้วยลูป while นี้เอง จึงมีความเป็นไปได้ว่าชุดคำสั่งภายในลูปอาจมิได้ถูกประมวลผลเลยก็ได้ หากเงื่อนไขที่ตรวจสอบครั้งแรกมีค่าเป็นเท็จ

2. รูปแบบคำสั่ง while

รูปแบบการเขียนคำสั่ง while แสดงดังต่อไปนี้

```
while(condition)
    {
        statement_1;
        statement_2;
        statement_3;
        ...
        statement_n;
    }
```

condition : เงื่อนไข หรือนิพจน์ที่สร้างขึ้น
statement : ประโยคชุดคำสั่งที่จะให้ทำงาน

ตัวอย่างการสร้างลูป while เช่น สร้างลูปด้วย while โดยจะให้ทำงานภายในลูปต่อเมื่อค่า a มีค่าน้อยกว่าหรือเท่ากับ 99 และหาก a มีค่ามากกว่า 99 เมื่อใด ก็จะหลุดออกจากลูป

```
while(a<=99)
```

สร้างลูปด้วย while โดยจะทำงานภายในลูปต่อเมื่อค่า x มีค่ามากกว่าหรือเท่ากับ 20 และค่า y มีค่าน้อยกว่าหรือเท่ากับ 40 และจะหลุดออกจากลูปต่อเมื่อเงื่อนไขใดเงื่อนไขหนึ่งเป็นเท็จ

```
while(x>=20 && y<=40)
```

ตัวอย่าง โปรแกรมแสดงคำว่า Navamin! จำนวน N บรรทัด ค่า N ถูกป้อนโดยผู้ใช้

```
#include <stdio.h>
main()
{
        int i,N;
        printf("Enter number for show Navamin : ");
        scanf("%d",&N);
        i = 1;                  //กำหนดรอบเริ่มต้นให้กับตัวแปร i
        while (i <= N)
        {
            printf("Navamin!\n");
             i = i + 1;         // เพิ่มรอบการทำงานให้ตัวแปร i
        }
}
```

เมื่อสั่งรันโปรแกรมจะปรากฏข้อความขึ้นมาให้เราป้อนตัวเลข ดังนี้

```
Enter number for show Navamin : _
```

เมื่อป้อนตัวเลขในกรณีที่เงื่อนไขของคำสั่ง while เป็นจริง จะปรากฏผลดังต่อไปนี้

```
Enter number for show Navamin : 6
Navamin!
Navamin!
Navamin!
Navamin!
Navamin!
Navamin!
```

ตัวอย่าง โปรแกรมที่ใช้การวนลูปด้วยคำสั่ง while เพื่อแสดงตัวเลข 0-10 ออกทางหน้าจอ

```
#include <stdio.h>
main()
{
        int count = 0       // สร้างตัวแปร count พร้อมกำหนดค่าเริ่มต้นให้เป็น 0
        printf("Show number from 0 to 10\n");
        while (count <= 10) // กำหนดเงื่อนไขให้วนทำงานในลูป ถ้าค่าของ count น้อยกว่าหรือเท่ากับ10
        {
        printf("%d",count);   // แสดงค่าของตัวแปร count ออกทางหน้าจอ
        count++;   //เพิ่มค่าของตัวแปร count ขึ้น 1 ก่อนที่จะวนไปตรวจสอบเงื่อนไขในรอบใหม่
        }
}
```

เมื่อสั่งรันโปรแกรมจะปรากฏผลการทำงาน ดังนี้

```
Show number from 0 to 10
0  1  2  3  4  5  6  7  8  9  10
```

ตัวอย่าง โปรแกรมบวกเลขโดยใช้ลูป while โดยเริ่มบวกจากเลข 0 ไปจนถึงตัวเลขที่ผู้ใช้กำหนด ยกตัวอย่างเช่น ถ้าผู้ใช้ป้อนค่าเป็น 10 โปรแกรมจะบวกเลขตั้งแต่ 0-10 (0+1+2+3+4+5+6+7+8+9+10)

```
#include<stdio.h>
main()
{
        int begin=0, sum = 0, end;
        printf("Enter end number : ");      // แสดงข้อความให้ป้อนตัวเลข
        scanf("%d", &end);   // รับตัวเลขเข้ามาเก็บไว้ในตัวแปร end
        while(begin <= end) // กำหนดเงื่อนไขให้วนทำงานจนกว่าค่าของตัวแปร begin จะเท่ากับ end
        {
        sum = sum + begin;    // สั่งให้บวกค่าของตัวแปร begin เข้ากับตัวแปร sum
        begin++;    // เพิ่มค่าของตัวแปร begin ขึ้นทีละหนึ่ง ก่อนที่จะวนทำงานในรอบต่อไป
        }
        printf("Sum = %d",sum); // เมื่อจบการวนลูป ให้แสดงค่าตัวแปร sum ออกทางหน้าจอ
}
```

ข้อควรระวังในการใช้คำสั่งลูปแบบ while ก็คือ ภายในลูปจะต้องมีหนึ่งคำสั่งในการเพิ่มค่าหรือลดค่า หรือเปลี่ยนแปลงค่าของตัวแปรที่ใช้ตรวจสอบเงื่อนไข เพื่อให้เงื่อนไขเข้าใกล้จุดที่จะเป็นเท็จลูปจะได้มีจุดจบ เพราะไม่เช่นนั้นจะกลายเป็นลูปที่ไม่มีวันจบ โปรแกรมจะวนทำงานไม่หยุดและไม่สามารถทำงานในคำสั่งต่อจากลูป while ได้

สรุป ภาษาซีมีชุดคำสั่งควบคุมแบบวนซ้ำอยู่ 3 ประเภท ได้แก่ คำสั่ง while, do while และ for โดยการควบคุมแบบวนซ้ำด้วยคำสั่ง while จะทำการตรวจสอบเงื่อนไขก่อนดำเนินการเสมอ ดังนั้น ชุดคำสั่งภายในลูปอาจมิได้ถูกประมวลผลเลยก็ได้ หากตรวจสอบเงื่อนไขครั้งแรกแล้วมีค่าเป็นเท็จ

กิจกรรมที่ 5.1 คำสั่ง while (คะแนนเต็ม 5 คะแนน) <ใช้เวลาทำ 10 นาที>

คำชี้แจง

ให้นักเรียนตอบคำถามต่อไปนี้

1. เขียนผังงาน (Flowchart) แสดงการทำงานของคำสั่ง while (1 คะแนน)

2. ถ้าต้องการสร้างลูปด้วย while โดยกำหนดให้ทำงานภายในลูปก็ต่อเมื่อค่า A มีค่ามากกว่าหรือเท่ากับ 100 และค่า B มีค่าน้อยกว่าหรือเท่ากับ 200 จะต้องเขียนคำสั่งอย่างไร (1 คะแนน)

.....

.....

3. เขียนโปรแกรมที่ใช้การวนลูปด้วยคำสั่ง while สร้างตัวนับถอยหลัง (เพื่อปล่อยจรวดไปยังดวงจันทร์) จากตัวเลข 10 ลงมายัง 0 ออกทางหน้าจอ (3 คะแนน)

ตัวอย่างผลลัพธ์

Show number from 10 to 0

10 9 8 7 6 5 4 3 2 1 0

.....

.....

.....

.....

.....

.....

.....

.....

.....

แนวตอบกิจกรรมที่ 5.1

คำสั่งวนลูปแบบ while จะเริ่มต้นทำงานจากการตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริง จึงจะทำงานตามคำสั่งของ while เมื่อทำงานเสร็จแล้วก็จะวนกลับไปตรวจสอบเงื่อนไขใหม่ เป็นเช่นนี้ไปเรื่อย ๆ จนกว่าเงื่อนไขจะเป็นเท็จจึงจะหลุดออกจากการทำงานของลูป while

ตัวอย่างการสร้างลูป while เช่น สร้างลูปด้วย while โดยจะให้ทำงานภายในลูปต่อเมื่อค่า x มีค่ามากกว่า 20 จะได้ while(x>20)

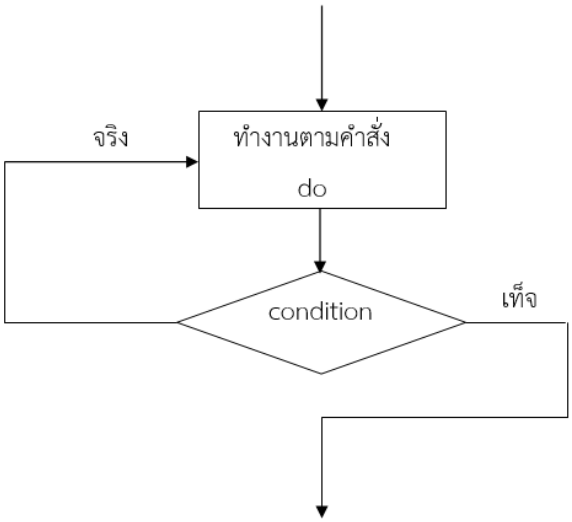
วิธีการนับถอยหลัง ใช้การกำหนดค่าเริ่มต้นด้วยค่าสูงสุด แล้วอาศัยการลดค่าลงครั้งละ 1 จนมีค่าเป็น 0 โดยถ้าต้องการลดค่า n ลงครั้งละ 1 ในภาษาซี สามารถใช้คำสั่ง n=n-1; หรือ n--; ได้

หัวเรื่องที่ 5.2 คำสั่ง do while

- เวลาเรียน 20 นาที -

1. ลักษณะการทำงานของคำสั่ง do while

คำสั่งวนลูปแบบ do-while จะคล้ายกับคำสั่ง while แต่จะต่างกันตรงที่ว่า ลูป do-while จะทำงานตามคำสั่งของ do ก่อนหนึ่งรอบแล้วจึงตรวจสอบเงื่อนไขที่ while ถ้าเงื่อนไขเป็นจริงจึงจะวนกลับไปทำงานตามคำสั่งของ do อีกครั้ง แล้วกลับมาตรวจสอบเงื่อนไขใหม่ เป็นอย่างนี้ไปจนกว่าผลการตรวจสอบเงื่อนไขจะเป็นเท็จ จึงจะเลิกทำงาน ออกจากลูป do-while สามารถสรุปการทำงานของลูป while ตามผังงาน (Flowchart) ดังรูปต่อไปนี้



2. รูปแบบคำสั่ง do while

รูปแบบการเขียนคำสั่ง do while แสดงดังต่อไปนี้

```
do
{
    statement_1;
    statement_2;
    statement_3;
    ...
    statement_n;
}
while(condition)
```

statement : ประโยคชุดคำสั่งที่จะให้ทำงาน
condition : เงื่อนไขหรือนิพจน์ที่ตรวจสอบเงื่อนไข

ตัวอย่าง โปรแกรมแสดงตัวเลขจาก 10 ลงมายัง 0 โดยใช้ลูปแบบ do-while

```
#include<stdio.h>
main()
{
        int num = 10;  // สร้างตัวแปร num พร้อมกำหนดค่าเริ่มต้นเป็น 10
        do                  // ใช้คำสั่งลูป do-while
        {
            printf("%d",num);       // แสดงค่าของตัวแปร num
            num--;                  // ลดค่าของตัวแปร num ลงหนึ่งก่อนออกจากลูป
        }
        while (num >= 0);   // ถ้าค่าของ num มากกว่าหรือเท่ากับ 0 จะวนกลับไปทำงานในรอบใหม่
}
```

เมื่อสั่งรันโปรแกรมจะปรากฏผลการทำงานดังนี้

```
10 9 8 7 6 5 4 3 2 1 0
```

ตัวอย่าง การเขียนโปรแกรมโดยใช้ลูป do-while แต่กำหนดเงื่อนไขของ while ให้เป็นเท็จตั้งแต่ต้น เพื่อดูผลการทำงานของโปรแกรม

```
#include <stdio.h>
main()
{
        int num = -20;       // กำหนดให้ตัวแปร num เก็บค่าเริ่มต้นเป็น -20
        do                   // ใช้คำสั่ง do-while
        {
        printf("%d",num);   // ให้แสดงค่าของตัวแปร num ออกทางหน้าจอ
        num--;              // ลดค่าของตัวแปร num ลงหนึ่ง num จึงมีค่า -19 หลังจากวนเข้ามารอบแรก
        }
        while (num >= 0);   // ตรวจสอบเงื่อนไข num ต้องมากกว่าหรือเท่ากับ 0 ทำให้เงื่อนไขเป็นเท็จ
}
```

เมื่อสั่งรันโปรแกรมจะปรากฏผลการทำงานดังนี้

```
-20
```

ผลการทำงานของโปรแกรมจะแสดงตัวเลข -20 ออกมาทางหน้าจอเพียงครั้งเดียว หลังจากนั้น จะออกจากลูป do-while ทันที โดยไม่มีการวนกลับไปทำงานตามคำสั่งของ do อีก เนื่องจากเงื่อนไขเป็นเท็จ สรุปได้ว่าถ้าใช้คำสั่งลูป do-while จะต้องมีการทำงานตามคำสั่ง do อย่างน้อย 1 รอบคำสั่ง

ตัวอย่าง โปรแกรมหาค่าเฉลี่ยของตัวเลข 10 จำนวนที่รับเข้ามาจากผู้ใช้ผ่านทางคีย์บอร์ด โดยใช้คำสั่งวนลูป do-while

```
#include <stdio.h>
main()
{
        int count = 0;
        float  num, total=0, average=0;
        do          // ใช้ลูปแบบ do-while
        {
            printf("Enter number : ");      // แสดงข้อความเป็นตัวเลข
            scanf("%f", &num);      // รับตัวเลขเข้ามาเก็บไว้ในตัวแปร num
            total += num;           // บวกค่าของตัวแปร num ก่อนที่จะออกจากลูป
            count++;     // เพิ่มค่าของตัวแปร count ก่อนที่จะออกจากลูป
        }
        while (count < 10);   // ตรวจสอบเงื่อนไข ตัวแปร count จะต้องมีค่าน้อยกว่า 10
        average = total/10;   // เมื่อออกจากลูปแล้ว ให้หาค่าเฉลี่ยโดยหารตัวแปร total ด้วย 10
        printf("\n***Result***\n");
        printf("Average = %.2f\n",average);  // แสดงผลออกทางหน้าจอ
}
```

ผลการทำงานเมื่อสั่งรันโปรแกรม พร้อมทั้งทดลองป้อนตัวเลขเข้าไป 10 จำนวน แสดงดังรูปต่อไปนี้

```
Enter number : 35
Enter number : 47
Enter number : -50
Enter number : 91
Enter number : 72
Enter number : -20
Enter number : 9
Enter number : 6
Enter number : 34
Enter number : 21

***Result***
Average = 24.50
```

สรุป การควบคุมแบบวนซ้ำด้วยคำสั่ง do while ชุดคำสั่งภายในลูป อย่างน้อยจะต้องถูกทำงาน 1 รอบเสมอ ถึงแม้ว่าการตรวจสอบเงื่อนไขในรอบแรกจะเป็นเท็จก็ตาม เนื่องจากลูป do while จะประมวลผลชุดคำสั่งภายในลูปรอบแรกก่อนเสมอ แล้วจึงค่อยตรวจสอบเงื่อนไขถ้าเป็นเท็จก็จะหลุดออกจากลูป

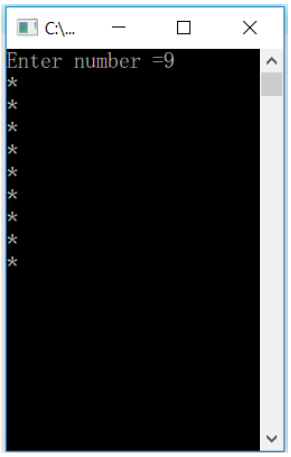
กิจกรรมที่ 5.2 คำสั่ง do while (คะแนนเต็ม 5 คะแนน) <ใช้เวลาทำ 10 นาที>

คำชี้แจง

ให้นักเรียนตอบคำถามต่อไปนี้

1. จงเขียนโปรแกรมแสดงผลรูป * เป็นแถวตามจำนวนตัวเลขที่ป้อนเข้าไปโดยใช้คำสั่ง do..while (5 คะแนน)

ตัวอย่างผลลัพธ์



แนวตอบกิจกรรมที่ 5.2

แนวคิดการเขียนโปรแกรมเพื่อแก้ปัญหาโจทย์

รับจำนวนแถวที่ต้องการจากแป้นพิมพ์ (คำสั่ง scanf)

```
do
{
    แสดงผลสิ่งที่ต้องการ  (*) ด้วยคำสั่ง printf
    count++;     // เพิ่มค่าของตัวแปร count ก่อนที่จะออกจากลูป
}
while (count < จำนวนแถวที่รับเข้ามาจากแป้นพิมพ์);
```

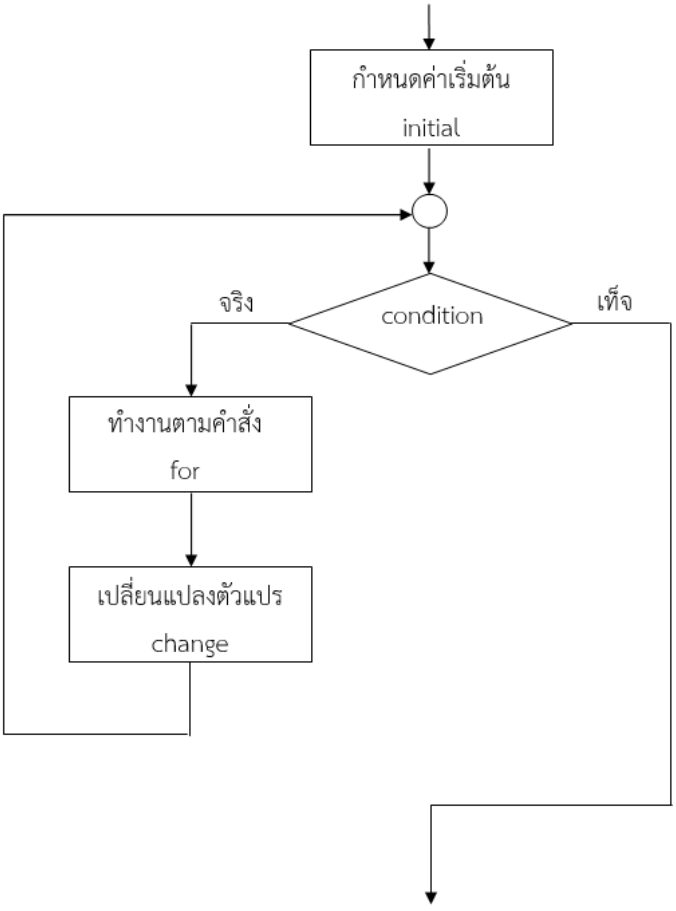
หัวเรื่องที่ 5.3 คำสั่ง for

- เวลาเรียน 20 นาที -

คำสั่ง for ใช้สำหรับการควบคุมทิศทางของโปรแกรมให้ทำงานแบบวนรอบ เช่นเดียวกับคำสั่ง while และ do-while แต่คำสั่ง for มีลักษณะพิเศษกว่าคำสั่งลูปแบบอื่น ๆ ตรงที่ คำสั่ง for เหมาะกับกรณีที่รู้จำนวนแน่นอนแล้วว่าต้องการให้วนลูปทำงานกี่รอบ

1. ลักษณะการทำงานของคำสั่ง for

คำสั่งวนลูปแบบ for จะเริ่มต้นทำงานจากการตรวจสอบค่าเริ่มต้นของตัวแปรที่นำมาใช้นับจำนวนรอบ ซึ่งอยู่ในส่วนแรกของคำสั่ง for รอบการทำงานซ้ำขึ้นอยู่กับนิพจน์เงื่อนไขที่ตั้งไว้ในส่วนที่ 2 ของคำสั่งและในส่วนที่ 3 เป็นการเพิ่มค่าตัวนับที่ส่งผลต่อจำนวนรอบของลูป สามารถสรุปการทำงานของลูป for ตามผังงาน (Flowchart) ดังรูปต่อไปนี้



2. รูปแบบคำสั่ง for

รูปแบบการเขียนคำสั่ง for แสดงดังต่อไปนี้

```
for (initial; condition; change)
{
    statement_1;
    statement_2;
    statement_3;
    ...
    statement_n;
}
```

initial : การกำหนดค่าเริ่มต้นของตัวแปรที่จะใช้เงื่อนไข ยกตัวอย่างเช่น x = 0 หมายถึง กำหนดตัวแปร x มีค่าเริ่มต้นเป็น 0

condition : เงื่อนไขที่กำหนดขึ้น เพื่อให้โปรแกรมวนทำงานตามคำสั่งหลัง for

change : ส่วนของการเปลี่ยนแปลงค่าของตัวแปรที่ใช้เป็นเงื่อนไข ซึ่งอาจจะเป็นการเพิ่มหรือลดค่าของตัวแปร (เช่น x = x-2)

statement : คำสั่งที่จะให้ทำงาน ถ้าผลการตรวจสอบเงื่อนไขออกมาเป็นจริง (True)

ตัวอย่าง โปรแกรมที่ใช้คำสั่ง for เพื่อวนลูปแสดงค่าของตัวอักษร a-z ออกทางหน้าจอ

```
#include <stdio.h>
main()
{
    char  letter;
    for(letter = 'a'; letter <= 'z';  letter++)
    printf("%c ",letter);
}
```

เมื่อสั่งรันโปรแกรมจะปรากฏผลการทำงานดังนี้

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

ตัวอย่าง โปรแกรมพิมพ์แม่สูตรคูณ 12 บรรทัด โดยรับค่าแม่สูตรคูณจากผู้ใช้ทางคีย์บอร์ด

```
#include <stdio.h>
main()
{
    int  i,n;
    printf("Enter multiply number :  ");
    scanf("%d",&n);
    for(i =1;i<=12;i++)
    printf("%d x %d = %d\n",n , i , n * i);
}
```

เมื่อรันโปรแกรม แล้วทดลองป้อนตัวเลขเท่ากับ 12 ปรากฏผลการทำงาน ดังนี้ (รอรับค่า n สมมติรับค่าสูตรคูณแม่ 12)

```
Enter multiply number :  12
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120
12 x 11 = 132
12 x 12 = 144
```

ตัวอย่าง โปรแกรมบวกเลขตั้งแต่ 1 ถึงค่า x โดยรับค่า x จากผู้ใช้ผ่านทางคีย์บอร์ด

```
#include <stdio.h>
main()
{
int  i,x,sum;
     sum=0;
     printf(“Enter last number = ”);
     scanf(“%d”,&x);
     for(i=1;i<=x;i++)
     sum = sum + i;
     printf(“Sum is = %d ”,sum);
}
```

เมื่อรันโปรแกรม แล้วทดลองป้อนตัวเลขเท่ากับ 30 ปรากฏผลการทำงานดังนี้ (แสดงค่าใน sum ซึ่งเป็นผลรวม)

```
Enter last number = 30
Sum is = 465
```

สรุป การควบคุมแบบวนซ้ำด้วยคำสั่ง for เป็นหนึ่งในชุดคำสั่งควบคุมแบบวนซ้ำของภาษาซีเหมาะกับกรณีที่การวนซ้ำมีจำนวนรอบการทำงานที่แน่นอน

กิจกรรมที่ 5.3 คำสั่ง for (คะแนนเต็ม 5 คะแนน) <ใช้เวลาทำ 10 นาที>

คำชี้แจง

ให้นักเรียนตอบคำถามต่อไปนี้

1. จงเขียนโปรแกรมหาค่า Factorial ของเลขจำนวนเต็มใดๆ (โดยใช้คำสั่ง for)

Factorial (!) คือการกระทำอย่างหนึ่งของคณิตศาสตร์ ผลของมันคือ การคูณจำนวนนับไล่ลงมาเรื่อยๆ จนถึงเลข 1 เช่น 5! (ห้าแฟกซ์) = 5x4x3x2x1 = 120 เป็นต้น (5 คะแนน)

แนวตอบกิจกรรมที่ 5.3

แนวคิดการเขียนโปรแกรมเพื่อแก้ปัญหาโจทย์

รับเลขจำนวนเต็มใดๆ (คำสั่ง scanf)

for(กำหนดค่าเริ่มต้น ; ตรวจสอบเงื่อนไข ; เพิ่มค่าการนับ)

{

fac=fac*จำนวนนับที่เปลี่ยนไปจนถึงค่าที่ป้อนเข้ามา;

}

แสดงผลลัพธ์ค่า Factorial

หัวเรื่องที่ 5.4 คำสั่งประกอบการควบคุมทิศทางแบบต่าง ๆ

- เวลาเรียน 25 นาที -

นอกจากคำสั่งควบคุมทิศทางการทำงานของโปรแกรมในแบบต่าง ๆ แล้ว ภาษาซียังมีคำสั่งอีกกลุ่มหนึ่ง ซึ่งอาจจะไม่ได้ใช้สำหรับควบคุมทิศทางการทำงานของโปรแกรมโดยตรง แต่คำสั่งเหล่านี้มักถูกนำไปใช้ร่วมกับคำสั่งอื่น ๆ เช่น นำไปใช้เพื่อหยุดการเลือกทำ หรือออกจากการทำงานของระบบ ซึ่งคำสั่งเหล่านี้ ได้แก่ คำสั่ง break, continue และ exit( )

1. คำสั่ง break

คำสั่ง break ถูกนำไปใช้ร่วมกับคำสั่งแบบเลือกทำ หรือวนรอบ เพื่อสั่งให้โปรแกรมหยุดการทำงานของคำสั่งแบบเลือกทำ หรือวนรอบ ที่กำลังทำงานอยู่ ตัวอย่างการใช้งานคำสั่ง break แสดงดังต่อไปนี้

ตัวอย่าง โปรแกรมที่มีการใช้คำสั่ง break ซ้อนอยู่ในคำสั่ง if

```
#include <stdio.h>
main()
{
int count = 0, num;
      while (count < 10)      {
     printf(“Enter number : ”);
     scanf(“%d”,&num);
     if (num == 0)              {
     printf(“Wrong number\n”);
     break;
     printf(“Enter new number\n”);
     }
     else
          printf(“Right number\n”);
     }
}
```

เมื่อสั่งรันโปรแกรม ให้ทดลองป้อนตัวเลขเข้าไปหลายๆ จำนวน จากนั้นให้ป้อนเลข 0 ซึ่งจะทำให้เงื่อนไขของคำสั่ง if เป็นจริง โปรแกรมจะแสดงข้อความ Wrong number แล้วออกจากการทำงานของคำสั่ง if และลูป while ทันที เนื่องจากคำสั่ง break ดังนั้น จึงไม่มีการแสดงข้อความ Enter new number ออกมาทางหน้าจอ

```
Enter number : 37
Right number
Enter number : 138
Right number
Enter number : -24
Right number
Enter number : 91
Right number
Enter number : 0
Wrong number
```

2. คำสั่ง continue

คำสั่ง continue จะใช้งานร่วมกับคำสั่งลูป (while, do-while และ for) เพื่อสั่งให้โปรแกรมหยุดการทำงานในรอบปัจจุบัน แล้ววนกลับไปเริ่มทำงานในรอบใหม่ ตัวอย่างการใช้งานคำสั่ง continue แสดงดังต่อไปนี้

ตัวอย่าง โปรแกรมที่ใช้คำสั่ง continue ซ้อนในลูป for

```
#include <stdio.h>
main()
{
    int count;
     for(count = 0; count <= 100; count++)
    {
        if(count % 10 == 0)
            printf("%d\n", count);
        else
    {
    continue;
    printf("The number can't be divided by 10.\n");
    }
    }
}
```

เมื่อสั่งรันโปรแกรม ถ้าเงื่อนไขของคำสั่ง if เป็นเท็จ โปรแกรมจะทำงานตามคำสั่ง continue นั่นคือหยุดการทำงานในรอบปัจจุบันกลับไปเริ่มทำงานในรอบใหม่ จึงทำให้ข้อความ The number can't be divided by 10. จะไม่ถูกแสดงขึ้นมาให้เห็นบนหน้าจอ ดังรูปต่อไปนี้

```
0
10
20
30
40
50
60
70
80
90
100
```

3. คำสั่ง exit( )

คำสั่ง exit() จะใช้สำหรับออกจากการทำงานของโปรแกรม รูปแบบการเรียกใช้คำสั่ง exit() แสดงดังต่อไปนี้

```
#include <stdlib.h>
...
exit(0);
```

#include <stdio.h> : พรีโพรเซสเซอร์ไดเร็คทีฟ เพื่อให้นำไฟล์ stdlib เข้ามารวม เนื่องจากในตัวโปรแกรมมีการเรียกใช้คำสั่ง exit()

exit(0) : ออกจากการทำงานของโปรแกรม โดยใส่เลขศูนย์ไว้ในวงเล็บหมายถึง การออกจากโปรแกรมแบบปกติ

ตัวอย่าง โปรแกรมหาผลหารของตัวเลข 2 จำนวน โดยที่มีการใช้คำสั่ง exit() เพื่อจบการทำงานของโปรแกรม ถ้าตัวหารที่รับเข้ามาเป็น 0

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    float x, y, z;
    printf("Enter 2 number for division x/y : ");
    scanf("%f/%f",&x, &y);
    if (y == 0)
    {
        printf("Error,divided by 0.\n");
        exit(0);
    }
    else        {
    z = x / y;
    printf("x divided by y = %.2f\n" ,z);
    }
}
```

เมื่อสั่งรันโปรแกรม ให้ทดลองป้อนตัวเลขเข้าไปในโปรแกรม โดยกำหนดตัวหารเป็น 0 เช่น 7/0 เพื่อดูผลการทำงานของโปรแกรม ดังแสดงในรูปต่อไปนี้

```
Enter 2 number for division x/y : 21/0
Error,divided by 0.
```

สรุป คำสั่งประกอบการควบคุมทิศทางการทำงาน เป็นคำสั่งที่ถูกนำไปใช้ร่วมกับคำสั่งอื่น ๆ เช่น นำไปใช้เพื่อหยุดการเลือกทำ หรือออกจากการทำงานของระบบ คำสั่งเหล่านี้ ได้แก่ คำสั่ง break สามารถนำมาใช้เพื่อสั่งให้หลุดออกจากลูปตามเงื่อนไขที่ได้กำหนดไว้ คำสั่ง continue นำไปใช้เพื่อสั่งให้วกกลับไปทำงานซ้ำที่ต้นลูปและ คำสั่ง exit() จะใช้สำหรับออกจากการทำงานของโปรแกรม

กิจกรรมที่ 5.4 คำสั่งประกอบการควบคุมทิศทางแบบต่าง ๆ (คะแนนเต็ม 5 คะแนน)

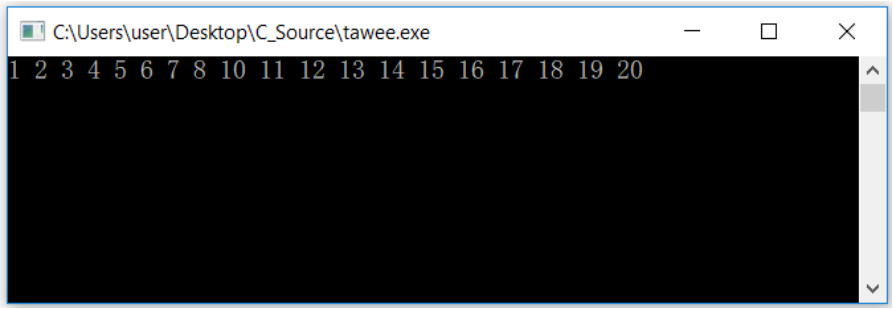
<ใช้เวลาทำ 15 นาที>

คำชี้แจง

ให้นักเรียนตอบคำถามต่อไปนี้

1. จงเขียนโปรแกรมแสดงผลตัวเลข 1-20 (ยกเว้นเลข 9) ออกทางจอภาพ โดยใช้คำสั่ง continue เป็นคำสั่งที่ให้โปรแกรม กลับไปทำงานที่ต้น loop โดยผลลัพธ์ที่ออกมา ไม่มีเลข 9 อยู่

ผลลัพธ์ของโปรแกรม



แนวตอบกิจกรรมที่ 5.4

แนวทางการเขียนโปรแกรม ต้องมีการตรวจสอบการแสดงผลว่าเริ่มจากเลข 1 ไปเรื่อยๆ แต่ถ้า (if) เมื่อใดที่ตรวจสอบว่าเป็นเลข 9 ก็ให้เลือกมาทำคำสั่ง continue