บทที่ 6

คำสั่งควบคุมแบบวนซ้ำ

(Repetition Statement)

ลูปคืออะไร

- ลูป (Loop) คือ โครงสร้างที่มีการย้อนกลับไปทำงานเดิมซ้ำอีกครั้ง เมื่อเงื่อนไขที่กำหนดมีค่าเป็นจริง

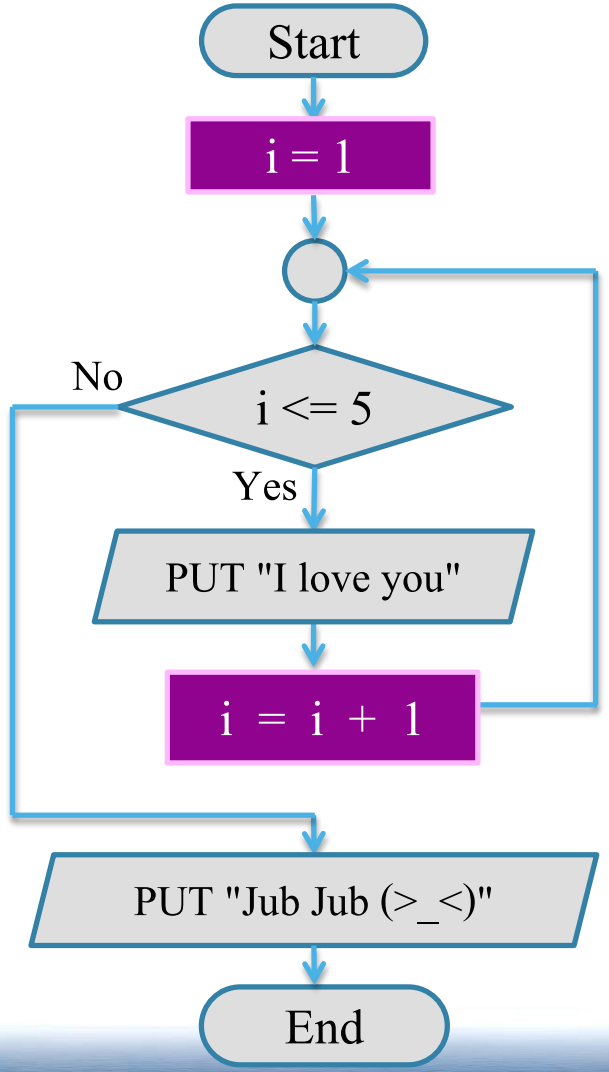
- ประโยชน์ของลูป คือ ช่วยลดบรรทัดของโค้ด เพราะเขียนชุดคำสั่งที่ต้องการทำซ้ำเพียงครั้งเดียว

ลูปเกิดในส่วนใดได้บ้าง

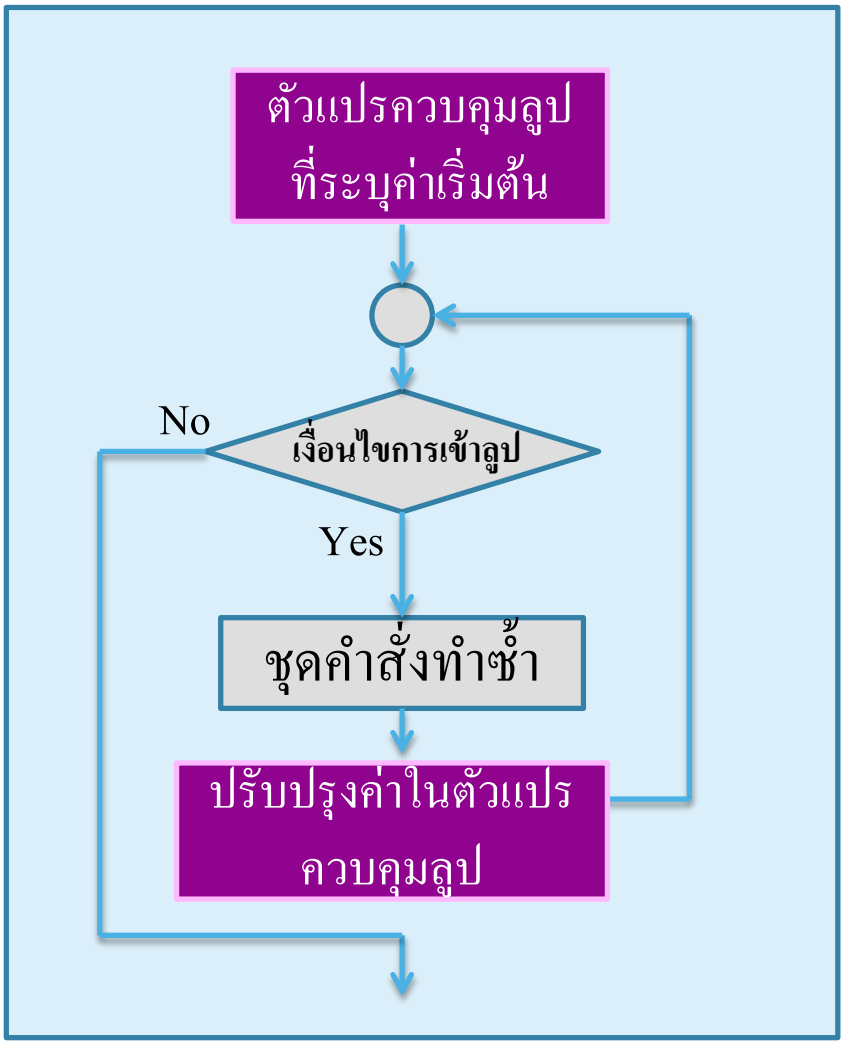
- Input
  - รับค่าซ้ำอีกครั้งเมื่อผู้ใช้กรอกค่าไม่ถูกต้อง
  - รับค่าตามจำนวนที่โปรแกรมต้องการ
- Processing
  - ประมวลผลซ้ำเพื่อนำผลลัพธ์ไปใช้ในรอบต่อไป
- Output
  - แสดงผลที่มีรูปแบบคล้ายคลึงกัน และมีความต่อเนื่อง

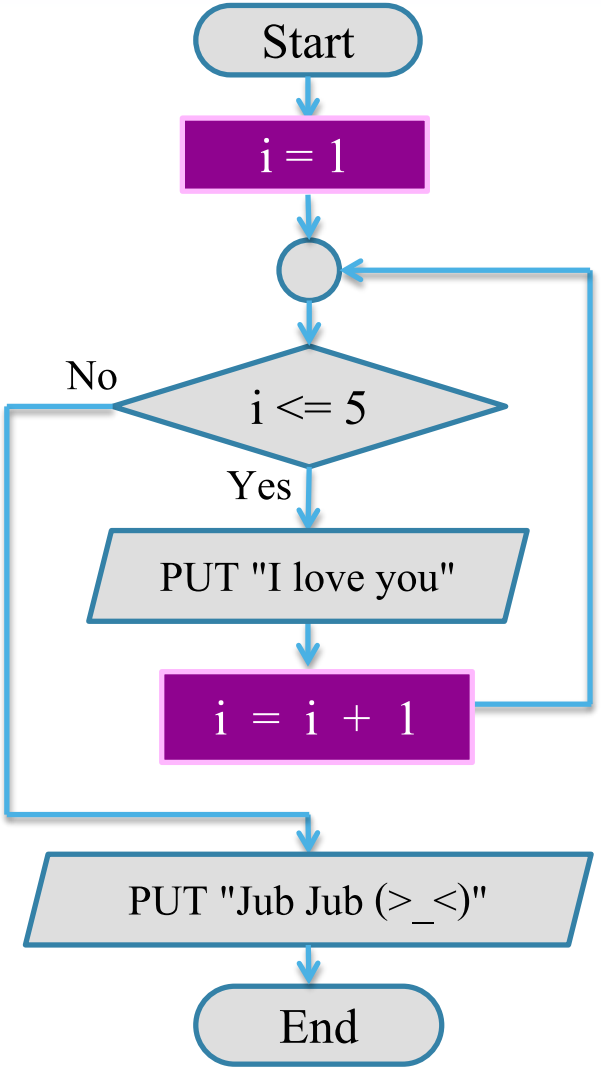
ตัวอย่างลูป

อัลกอริทึมของโปรแกรมบอกรักแฟน 5 ครั้ง



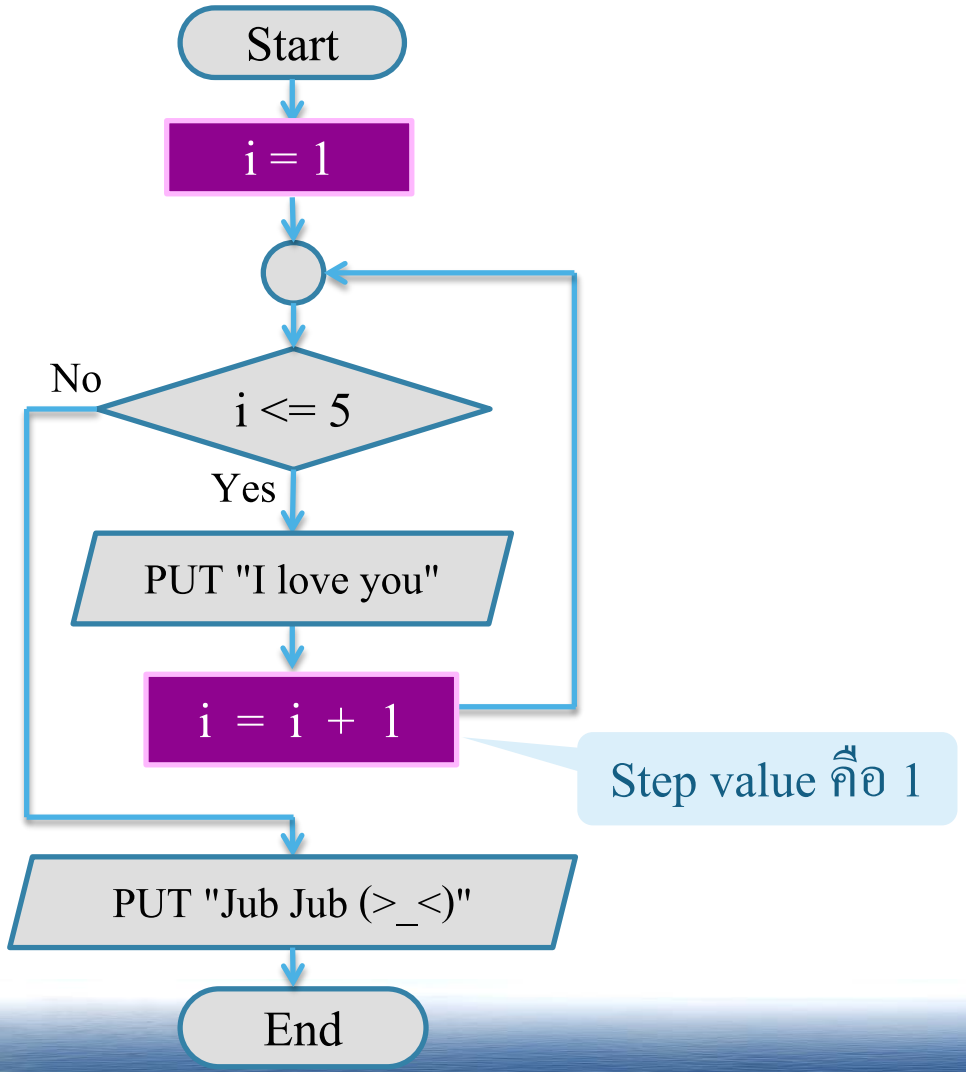
ลูปแบบจำกัดจำนวนรอบ

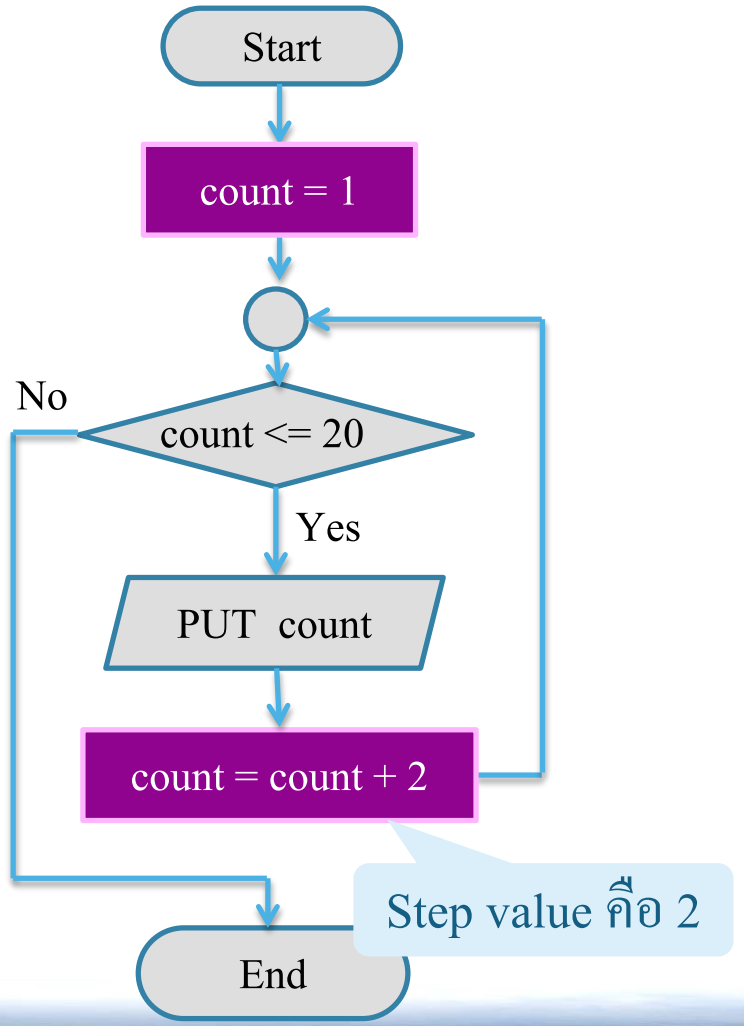




Step value

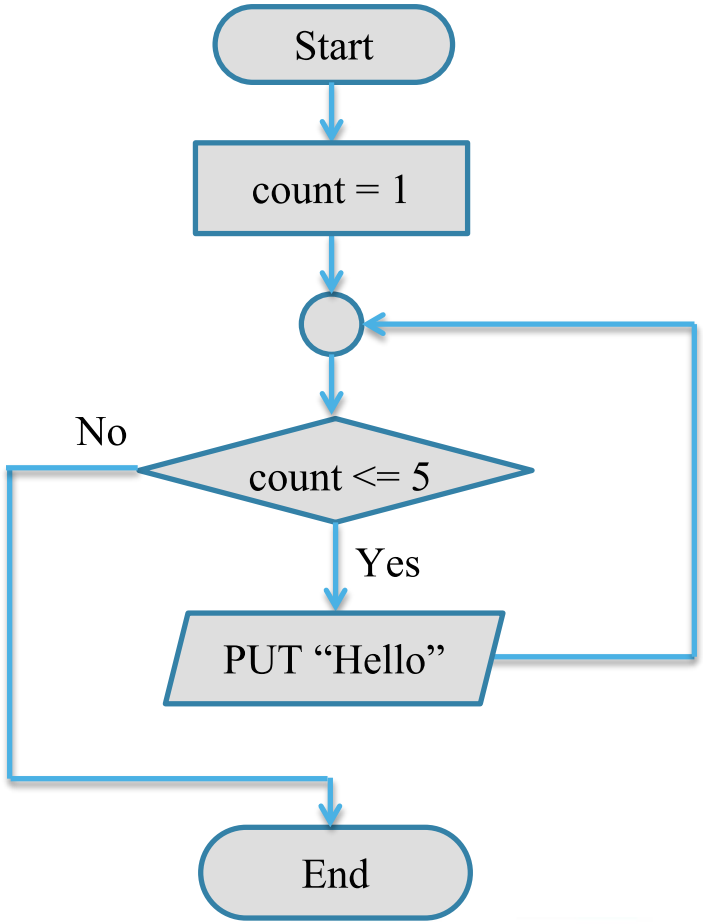
- Step value คือ ช่วงของค่าในตัวแปรควบคุมลูปแต่ละรอบ





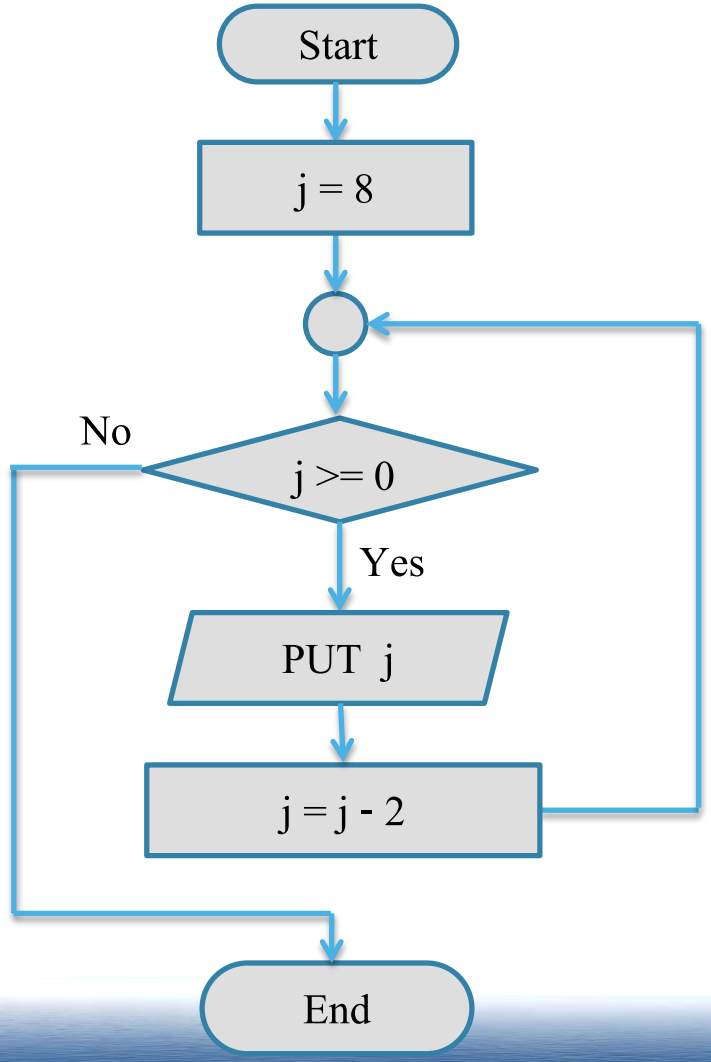
กิจกรรม

พิจารณาอัลกอริทึมของโปรแกรมแสดง Hello 5 ครั้งต่อไปนี้ ถูกหรือผิด หากผิดต้องแก้ไขส่วนใด



กิจกรรม

- จงเขียนค่าของตัวแปร j ในแต่ละรอบ และผลลัพธ์ที่ได้จากอัลกอริทึม



ตัวแปร

j = 8

Output

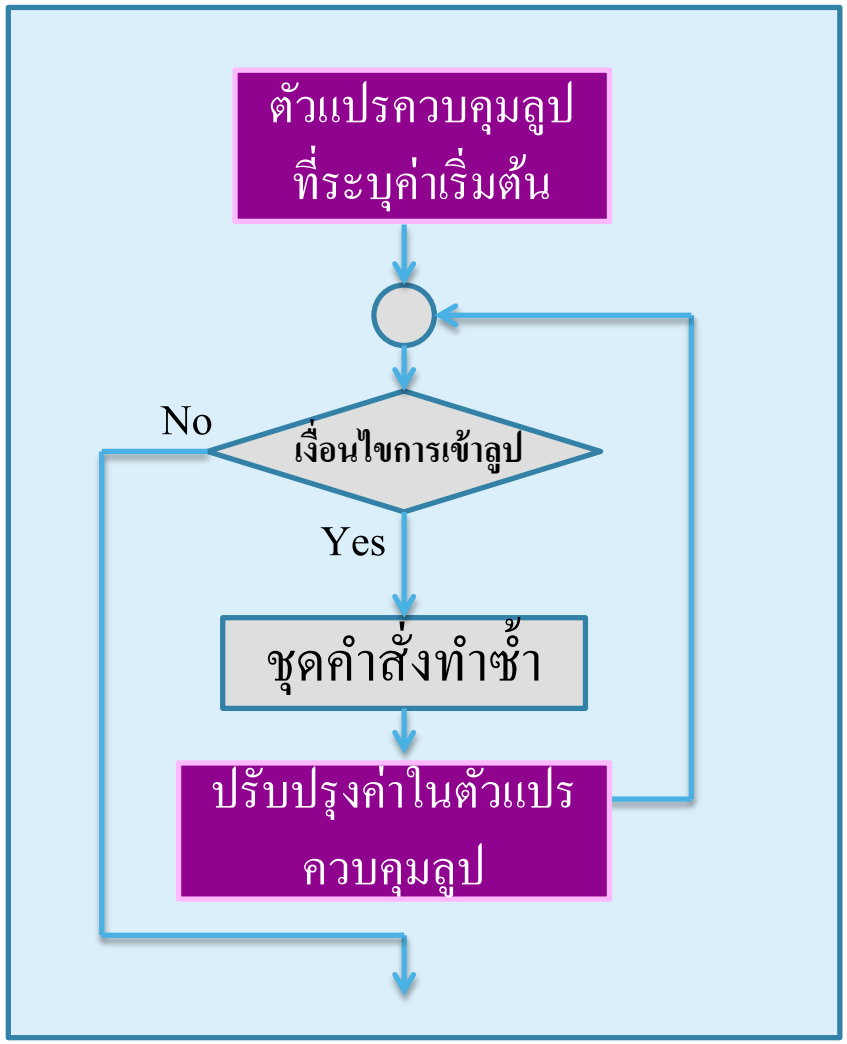
กิจกรรม

- จงเขียนอัลกอริทึม โปรแกรมวนลูปแสดงตัวเลขจาก 10 ถึง 1

ผลลัพธ์

10 9 8 7 6 5 4 3 2 1

รูปแบบคำสั่งลูป while

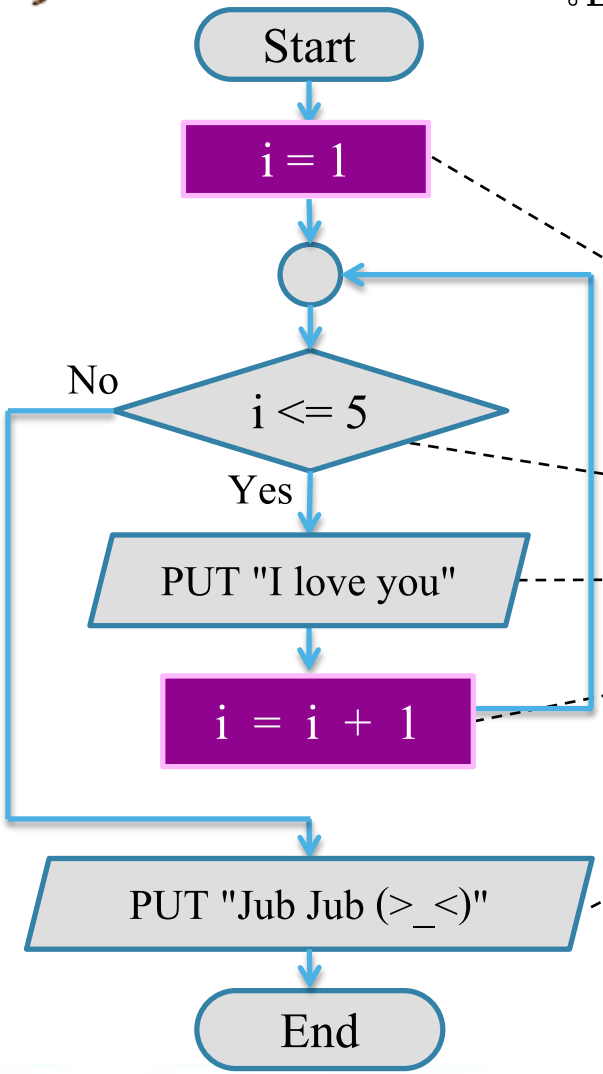


ตัวแปรควบคุมลูป ที่ระบุค่าเริ่มต้น

```
while( เงื่อนไขการเข้าลูป )
{
    ชุดคำสั่งทำซ้ำ
    ปรับปรุงค่าในตัวแปรควบคุมลูป
}
```

ตัวอย่าง

โปรแกรมบอกรักแฟน 5 ครั้ง




```
1  #include <stdio.h>
2
3  void main() {
4      int i = 1;
5      while (i <= 5) {
6          printf("I love you\n");
7          i++;
8      }
9      printf("Jub Jub (>_<)\n");
10 }
```

ตัวอย่าง

โปรแกรมบอกรักแฟน 5 ครั้ง

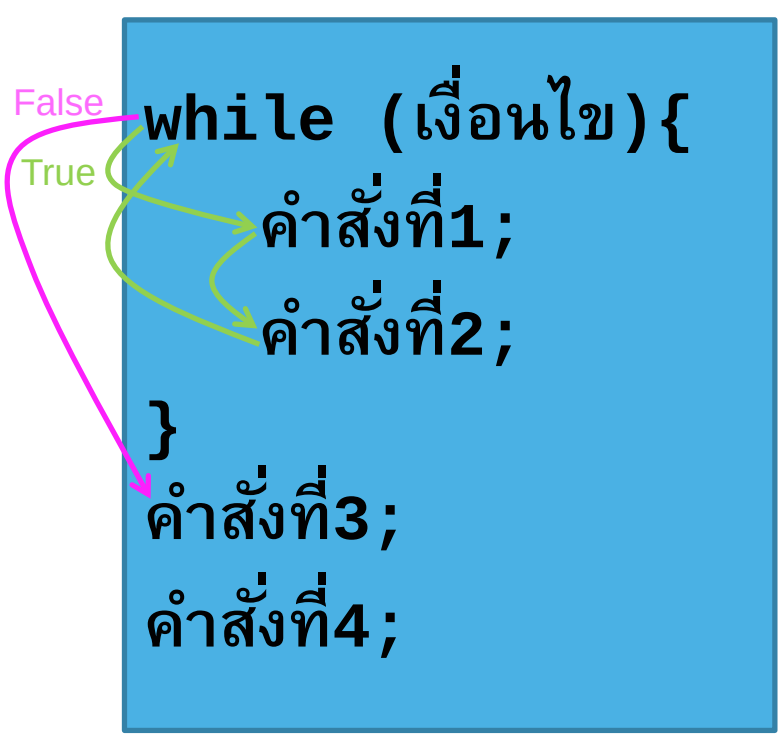
```
1  #include <stdio.h>
2
3  void main() {
4      int i = 1;
5      while (i <= 5) {
6          printf("I love you\n");
7          i++;
8      }
9      printf("Jub Jub (>_<)\n");
   }
```

Output

```
I love you
I love you
I love you
I love you
I love you
Jub Jub (>_<)
```

i = ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6

การใช้คำสั่ง while ควบคุมลูป



ลำดับการทำงาน

1. ทดสอบนิพจน์เงื่อนไข ซึ่งจะได้ผลลัพธ์เป็นจริง หรือ เท็จ
2. หากนิพจน์เป็นจริง จะทำคำสั่งที่อยู่ภายใต้ while ที่อยู่ในวงเล็บปีกกา { }
3. หลังจากนั้นจะย้อนกลับไปตรวจสอบเงื่อนไขใน while
4. หากนิพจน์เป็นเท็จ จะไม่สนใจคำสั่งภายใต้ while จะข้ามไปยังคำสั่งหลังปีกกาปิดของ while แต่หาเป็นจริงจะทำคำสั่งภายใต้วงเว็บปีกกาอีกครั้ง

กิจกรรม

- จากอัลกอริทึมโปรแกรมวนลูปแสดงตัวเลขจาก 10 ถึง 1 จงเขียนเป็นชุดคำสั่งภาษาซีด้วยคำสั่ง while

กิจกรรม

จงเขียนอัลกอริทึมและโค้ดของโปรแกรมแสดงผลคูณของค่าที่รับเข้ามา คูณกับตัวเลขตั้งแต่ 1 ถึง 12

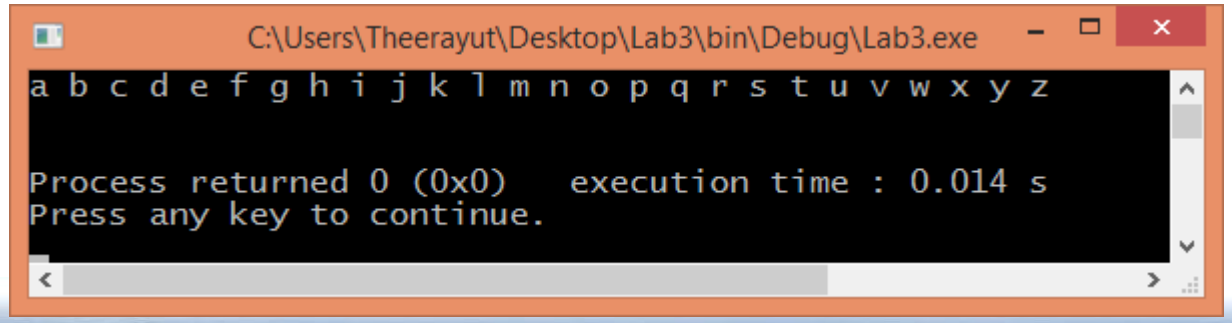
Test case:

Input	Output
5	5 x 1 = 5 5 x 2 = 10 5 x 3 = 15 5 x 4 = 20 5 x 5 = 25 5 x 6 = 30 5 x 7 = 35 5 x 8 = 40 5 x 9 = 45 5 x 10 = 50 5 x 11 = 55 5 x 12 = 60

ตัวแปรควบคุมลูปชนิด char

โปรแกรมแสดงตัวอักษรจาก a ถึง z โดยใช้คำสั่ง while

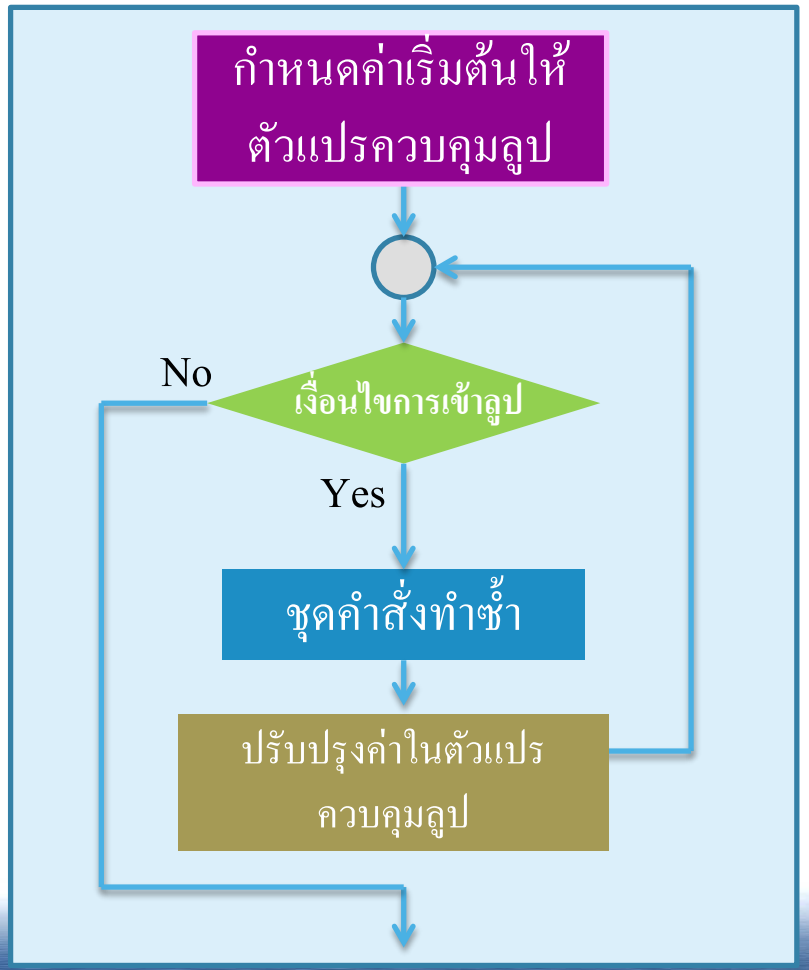
```
#include <stdio.h>

void main() {
    char ch = 'a';

    while (ch <= 'z') {
        printf("%c ", ch);
        ch = ch + 1;
    }
    printf("\n\n");
}
```



รูปแบบคำสั่งลูป for

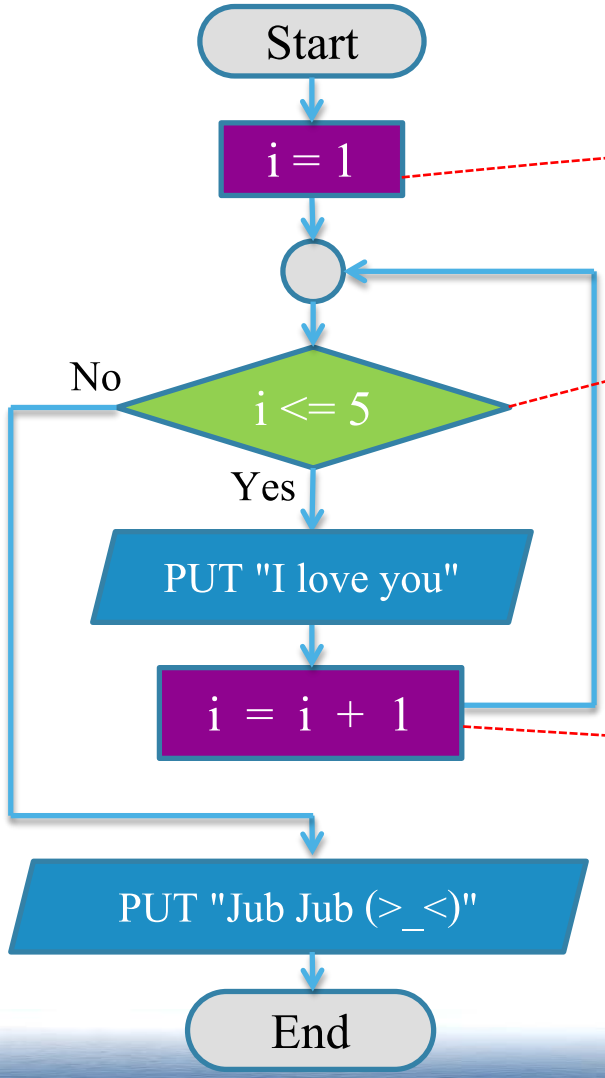
```
for( กำหนดค่าเริ่มต้นให้ตัวแปรควบคุม ; เงื่อนไขการเข้าลูป ; ปรับปรุงค่าในตัวแปรควบคุม )
{
        ชุดคำสั่งทำซ้ำ
}
```

- คำสั่งทำซ้ำนอกจากคำสั่ง while ยังสามารถใช้คำสั่ง for แทนได้
- คำสั่ง for 1 คำสั่ง จะบรรจุการทำงานไว้ 3 คำสั่ง ประกอบด้วย
  1) คำสั่งกำหนดค่าเริ่มต้นให้ตัวแปรควบคุมลูป
  2) เงื่อนไขการเข้าลูป
  3) คำสั่งปรับปรุงค่าในตัวแปรควบคุมลูป



ตัวอย่าง

โปรแกรมบอกรักแฟน 5 ครั้ง




```c
#include <stdio.h>

void main() {
    int i;
    for (i=1; i<=5; i++) {
        printf("I love you\n");
    }
    printf("Jub Jub (>_<)\n");
}
```

ตัวอย่าง

```
1  #include <stdio.h>
2
3  void main() {
4      int i;
5      for (i=1; i<=5; i++) {
6          printf("I love you\n");
7      }
8      printf("Jub Jub (>_<)\n");
   }
```

ชื่อตัวแปร \ รอบการทำงาน	ค่าเริ่มต้น	รอบที่ 1	รอบที่ 2	รอบที่ 3	รอบที่ 4	รอบที่ 5
i	1	2	3	4	5	6

Output

I love you
I love you
I love you
I love you
I love you
Jub Jub (>_<)

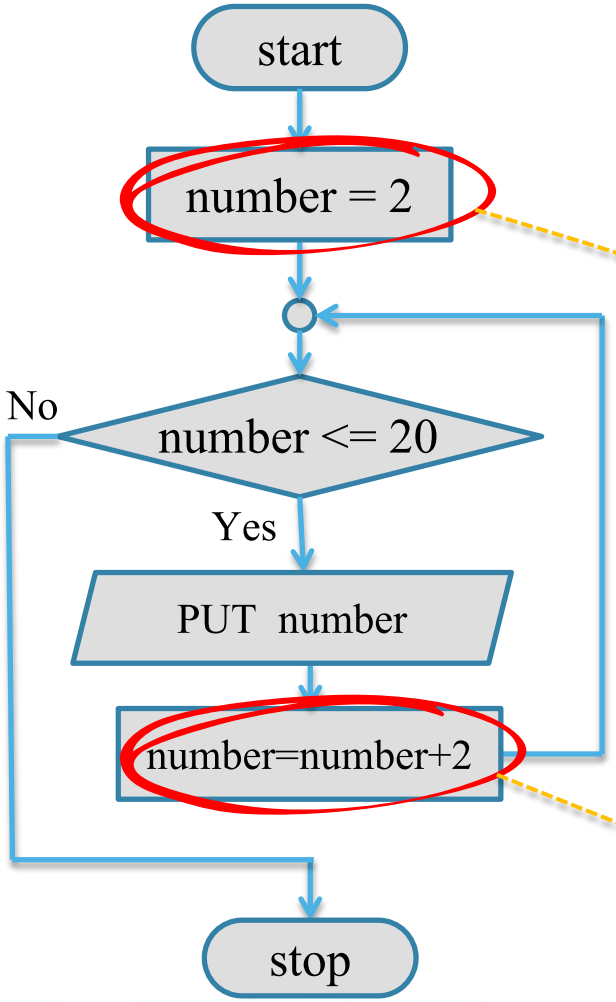
เปรียบเทียบคำสั่ง while และ for

```c
#include <stdio.h>

void main() {
    int i = 0;
    while (i < 5) {
        printf("I love you\n");
        i++;
    }
    printf("Jubu Jubu (>_<)\n");
}
```

```c
#include <stdio.h>

void main() {
    int i;
    for (i=0; i<5; i++) {
        printf("I love you\n");
    }
    printf("Jubu Jubu (>_<)\n");
}
```

ตัวอย่าง

โปรแกรมแสดงเลขคู่ในช่วง 1 ถึง 20




```
1  #include <stdio.h>
2
3  void main() {
4    int number;
5
6    for(number=2; number<=20; number+=2) {
7        printf("%d ", number);
8    }
9
10 }
```

เพิ่มค่า number ขึ้นทีละ 2

กิจกรรม

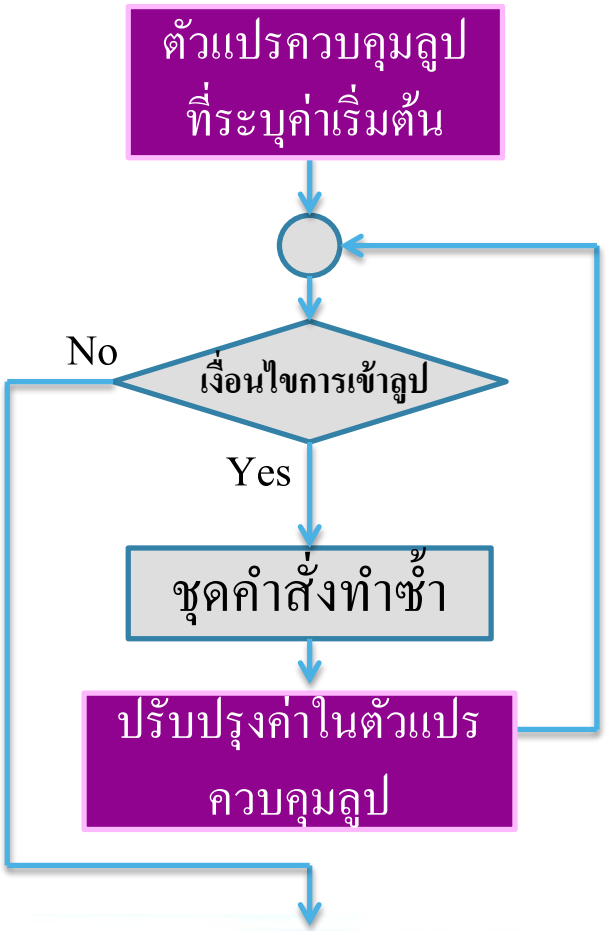
- จากอัลกอริทึมโปรแกรมวนลูปแสดงตัวเลขจาก 10 ถึง 1 จงเขียนเป็นชุดคำสั่งภาษาซีด้วยคำสั่ง for

กิจกรรม

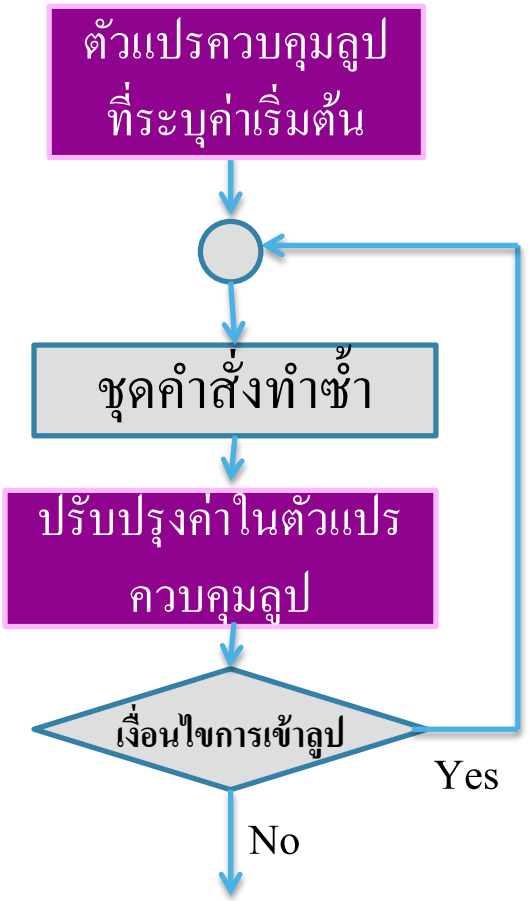
- จงเขียน Algorithm ในการวนลูปเพื่อแสดงตัวเลขคู่ ตั้งแต่ 1-100 ด้วย Raptor
- สร้างเป็นโปรแกรมด้วยภาษาซี โดยใช้คำสั่งทำซ้ำ while และ for

Pre-test Loop & Post-test Loop

Pre-test Loop



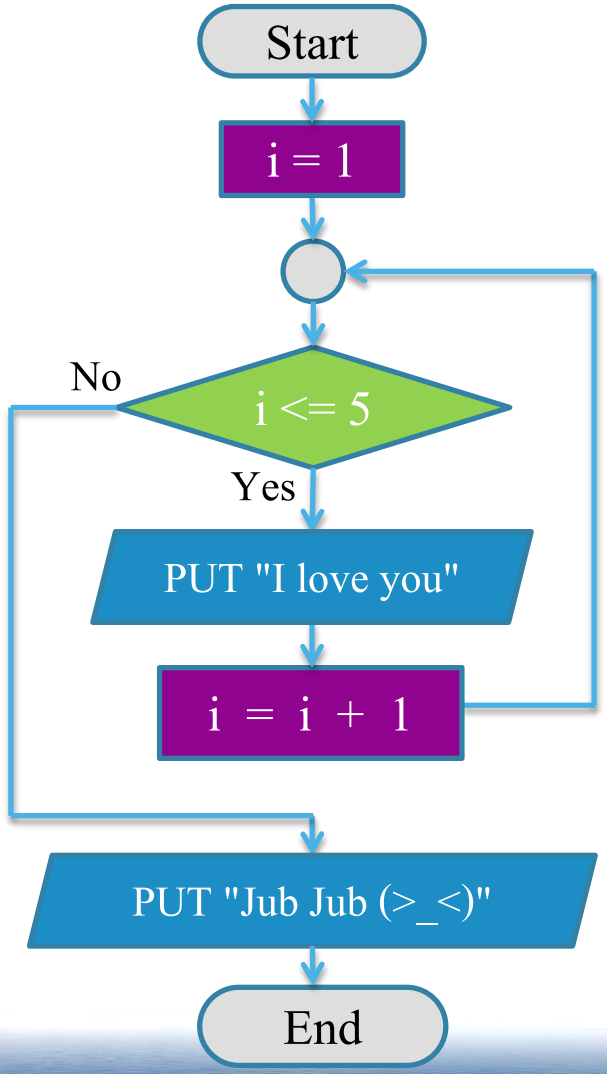
Post-test Loop



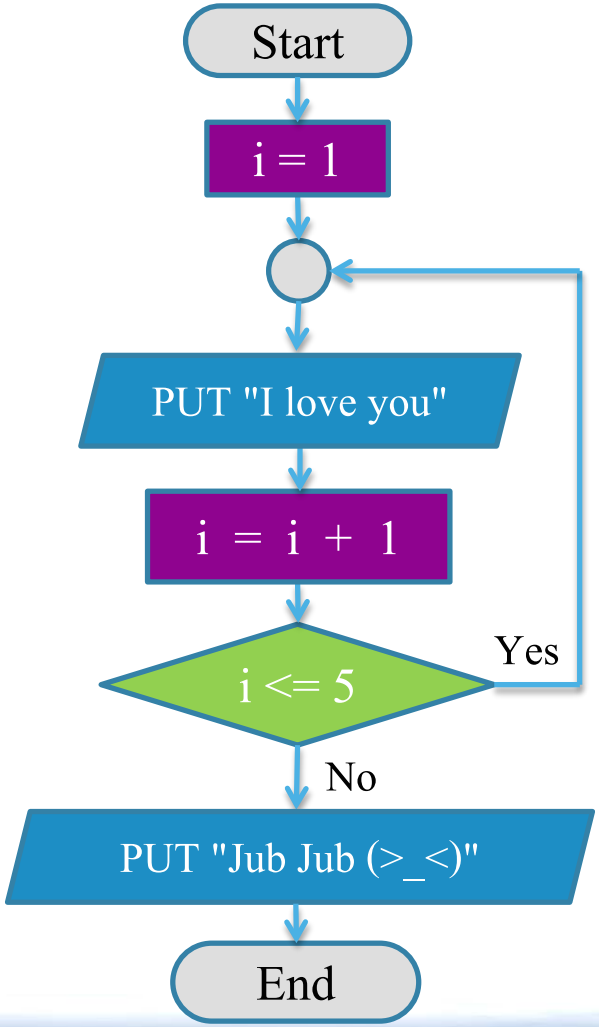
*โปรแกรม RAPTOR ไม่มี Post-test Loop

โปรแกรมบอกรักแฟน 5 ครั้ง

Pre-test Loop

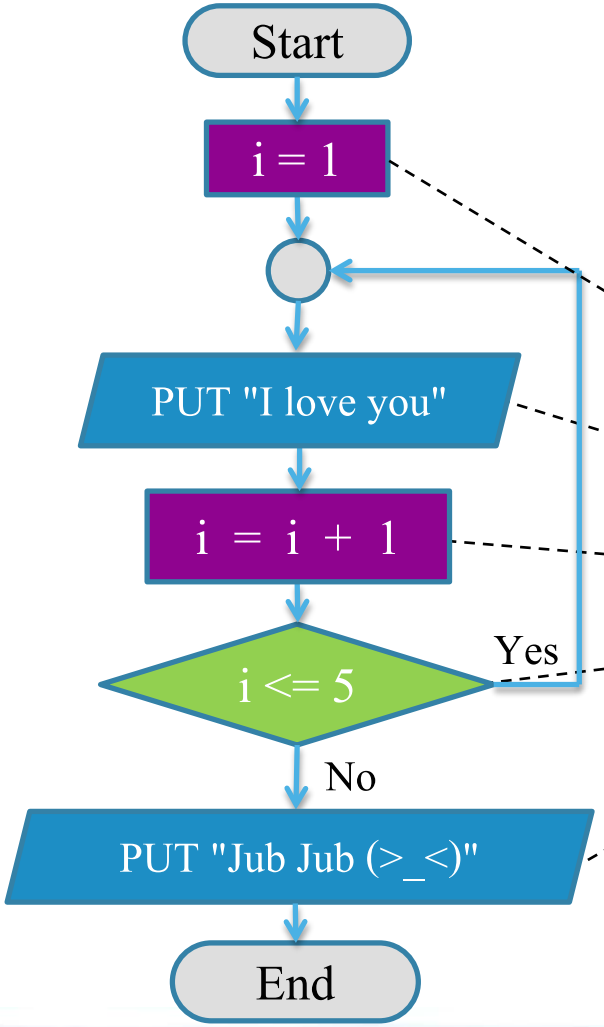


Post-test Loop



ตัวอย่าง

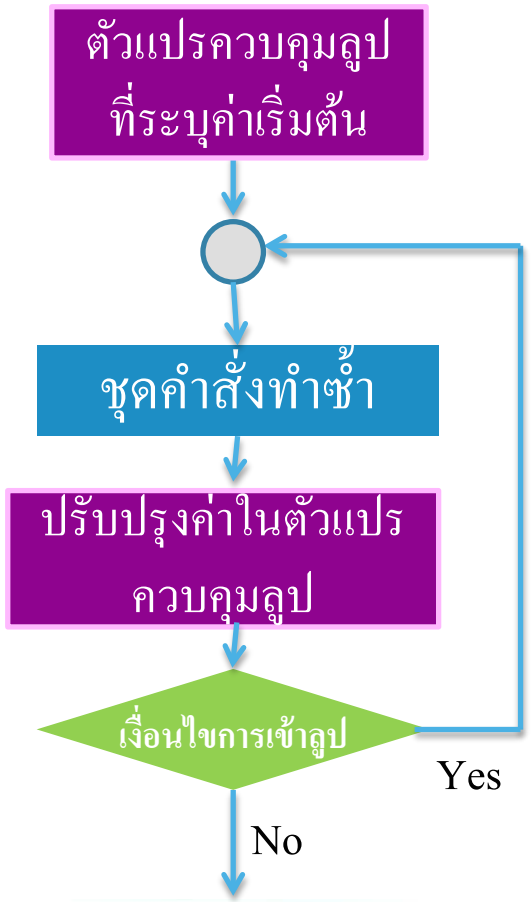
โปรแกรมบอกรักแฟน 5 ครั้ง




```
1  #include <stdio.h>
2
3  void main() {
4      int i = 1;
5      do {
6          printf("I love you\n");
7          i++;
8      } while (i <= 5);
9      printf("Jub Jub (>_<)\n");
10 }
```

รูปแบบคำสั่งลูป do-while

(Post-test Loop)



ตัวแปรควบคุมลูป ที่ระบุค่าเริ่มต้น

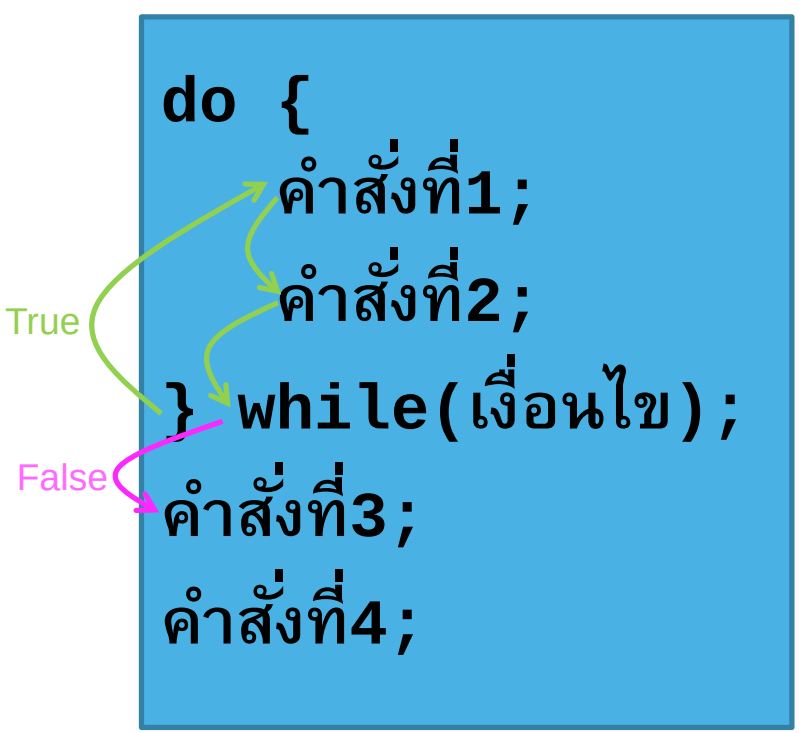
```
do {
```

ชุดคำสั่งทำซ้ำ

ปรับปรุงค่าในตัวแปรควบคุมลูป

```
} while(
```
เงื่อนไขการเข้าลูป
```
);
```

การใช้คำสั่ง do…while ควบคุมลูป



ลำดับการทำงาน

1. ทำชุดคำสั่งในวงเล็บปีกกาหลัง do ก่อนจนเสร็จ
2. หลังจากนั้นจะทดสอบนิพจน์เงื่อนไข ในบรรทัดคำสั่ง while ซึ่งจะได้ผลลัพธ์เป็นจริง หรือ เท็จ
3. หากนิพจน์เป็นจริง จะย้อนกลับไปทำชุดคำสั่งในวงเล็บปีกกาหลัง do อีกครั้ง แต่ถ้าเป็นเท็จจะทำคำสั่งที่อยู่ถัดจากคำสั่ง while

การใส่ { } ในคำสั่งลูป

กรณีที่คำสั่งภายในลูปมีเพียง 1 คำสั่ง ไม่ต้องใส่ปีกกาครอบก็ได้ แต่ถ้า 2 คำสั่งขึ้นไป ต้องใส่เสมอ

คำสั่งเดียว ไม่ต้องใส่ { } ก็ได้

```
while (เงื่อนไข)
    คำสั่ง;
```

```
do
    คำสั่ง;
while(เงื่อนไข);
```

```
for(…;…;…)
    คำสั่ง;
```

```
while (เงื่อนไข) {
    คำสั่งที่1;
    คำสั่งที่2;
}
```

```
do {
    คำสั่งที่1;
    คำสั่งที่2;
} while(เงื่อนไข);
```

```
for(…;…;…) {
    คำสั่งที่1;
    คำสั่งที่2;
}
```

หลายคำสั่ง ให้ใส่ { } เสมอ

การหยุด Loop ก่อนกำหนด

- ในระหว่างที่มีการทำซ้ำในลูป สามารถสั่งให้ลูปหยุดได้ ซึ่งประกอบด้วย 2 คำสั่ง
  - คำสั่ง break คือ คำสั่งที่ใช้เมื่อต้องการหยุดการทำงานของลูปทันที
  - คำสั่ง continue คือ คำสั่งที่ใช้เมื่อต้องการให้หยุดการทำงานลูปแล้วกลับไปตรวจสอบเงื่อนไขของลูป

คำสั่ง break

```
#include <stdio.h>
void main() {
    int i = 1;

    while(i<=5) {
        printf("OK\n");
        i++;
        if(i==3) {
            break;
        }
    }

    printf("End at i=%d\n", i);

}
```


```
OK
OK
End at i=3
```

หน้าจอ

คำสั่ง continue

```
#include <stdio.h>
void main() {

    int i;

    for (i=1; i<=4; i++) {
        if(i==3) {
            continue;
        }
        printf("%d\n", i);
    }

}
```

1
2
4

หน้าจอ

การใช้ลูปหาค่าผลรวม

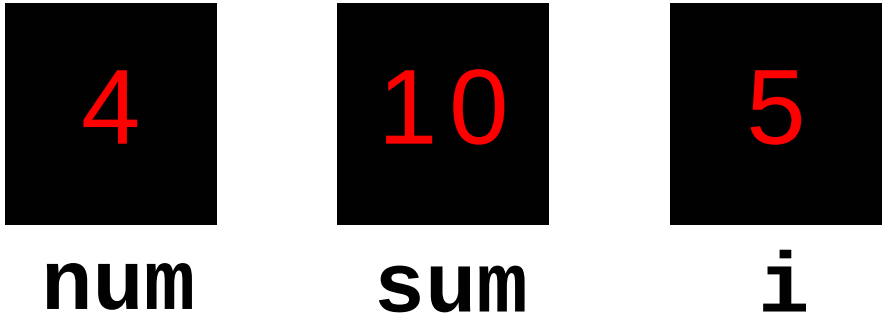
- การหาค่าผลรวม จะใช้ตัวแปรในการสะสมค่า ซึ่งเรียกว่า Accumulator

- Accumulator จะถูกกำหนดค่าใหม่ เมื่อจบการทำงานของลูปในแต่ละรอบ

ตัวอย่าง

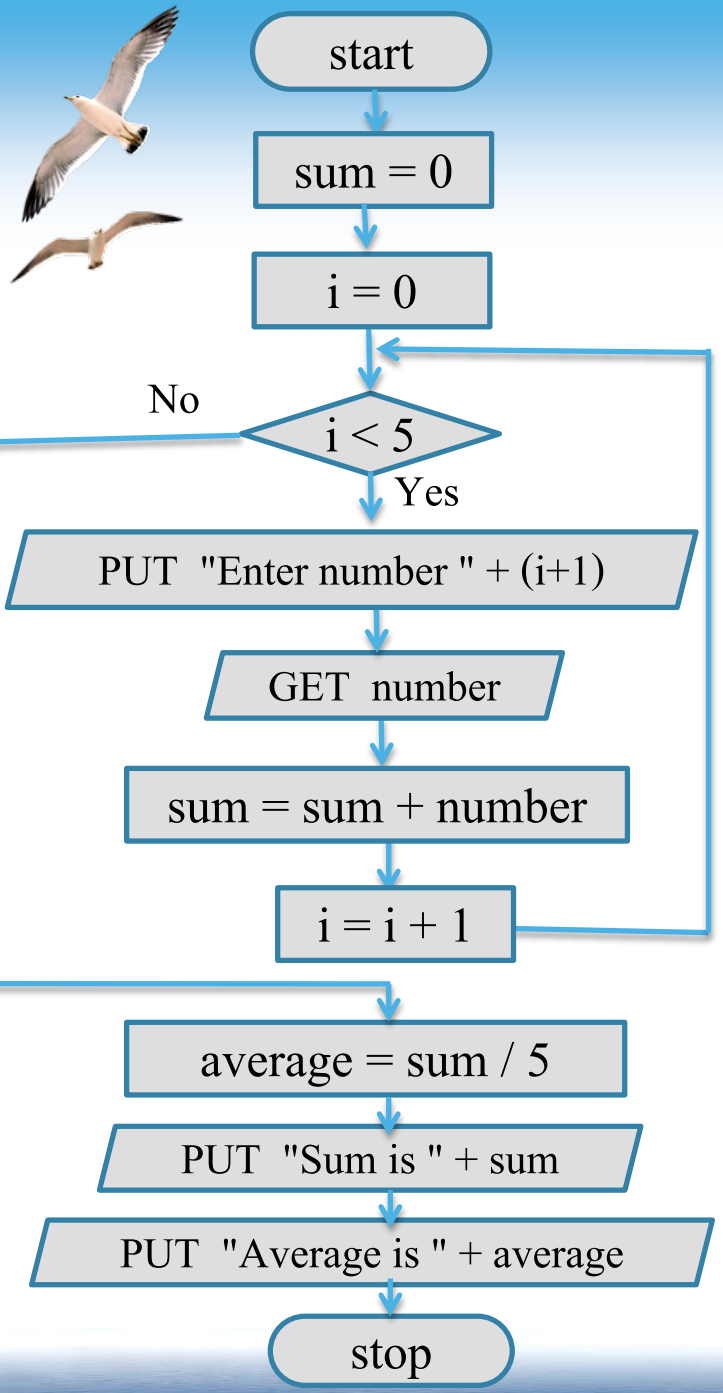
โปรแกรมบวกเลขตั้งแต่ 1 ถึงค่าที่ input จากผู้ใช้

```
#include <stdio.h>

void main() {
    int num, sum, i;
    printf("Input number = ");
    scanf("%d", &num);

    sum = 0;
    for (i=1; i<=num; i++) {
        sum = sum + i;
    }

    printf("sum = %d\n", sum);
}
```



หน้าจอ

```
Input number = 4↵

sum = 10
```



ตัวอย่าง

โปรแกรมรับเลขจำนวนเต็มจากผู้ใช้จำนวน 5 ค่า และหาว่าค่าเฉลี่ยของเลขที่ป้อนเข้ามาเป็นเท่าใด

```
1  #include <stdio.h>
2
3  void main() {
4      int i, number, sum = 0;
5      float average;
6
7      for (i=0; i<5; i++) {
8          printf("Enter number %d : ", i+1);
9          scanf("%d", &number);
10         sum = sum + number;
11     }
12
13     average = sum / 5.0;
14     printf("Sum is %d\n", sum);
15     printf("Average is %.2f", average);
16 }
```

ประกาศตัวแปร sum เพื่อใช้สะสมค่าผลบวก ต้องกำหนดค่าเริ่มต้นเสมอ

นำค่าปัจจุบันที่อยู่ในตัวแปร sum ไปบวกกับตัวแปร number และนำผลลัพธ์ไปทับค่าเดิมในตัวแปร sum

Debug

กิจกรรม

- จงเขียน Algorithm ในการหาค่าผลรวมของ

  1/1 + 1/2 + 1/3 + 1/4 + ... + 1/100

  ซึ่งมีผลลัพธ์เป็น 5.19

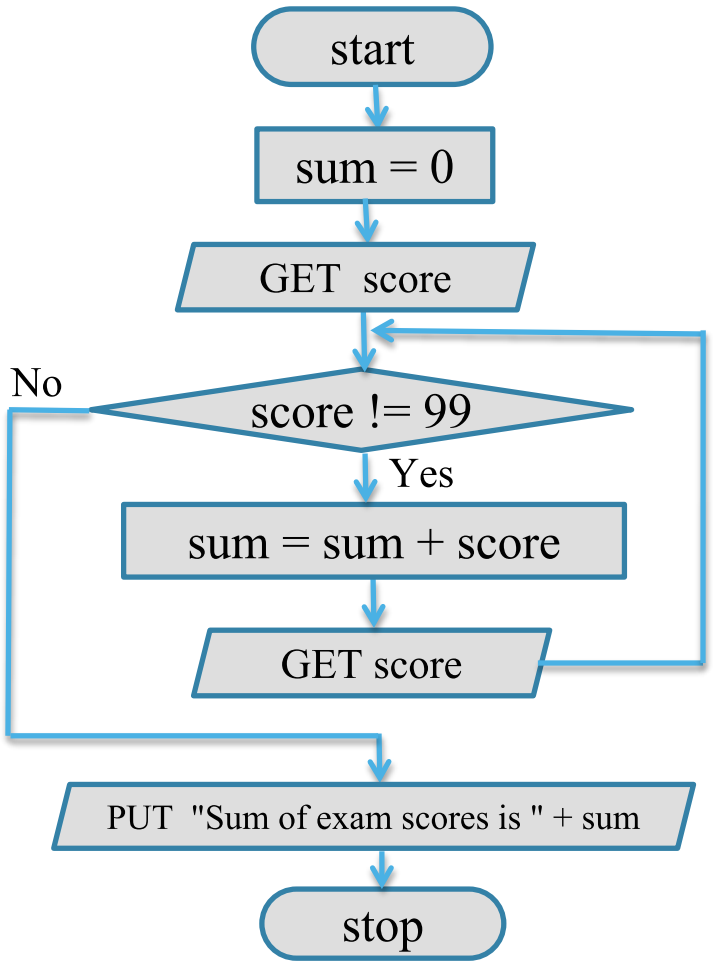
- สร้างเป็นโปรแกรมด้วยภาษาซี โดยใช้คำสั่งทำซ้ำ while

ประเภทของลูป

- **Definite Loop** (หรือ Counted Loop) คือ ลูปแบบจำกัดจำนวนรอบการทำงาน
  - ใช้ตัวแปรในการควบคุมจำนวนรอบของลูป

- **Infinite Loop** คือ ลูปแบบไม่จำกัดจำนวนรอบการทำงาน
  - ลูปควบคุมด้วยค่าเฝ้าดู (Sentinel-controlled loop) คือ ลูปที่จะหยุดเมื่อตัวแปรควบคุมลูปมีค่าตรงกับ "ค่าเฝ้าดู" (Sentinel value)
  - ลูปตรวจสอบความถูกต้องของ input (Input Validation)

ลูปควบคุมด้วยค่าเฝ้าดู

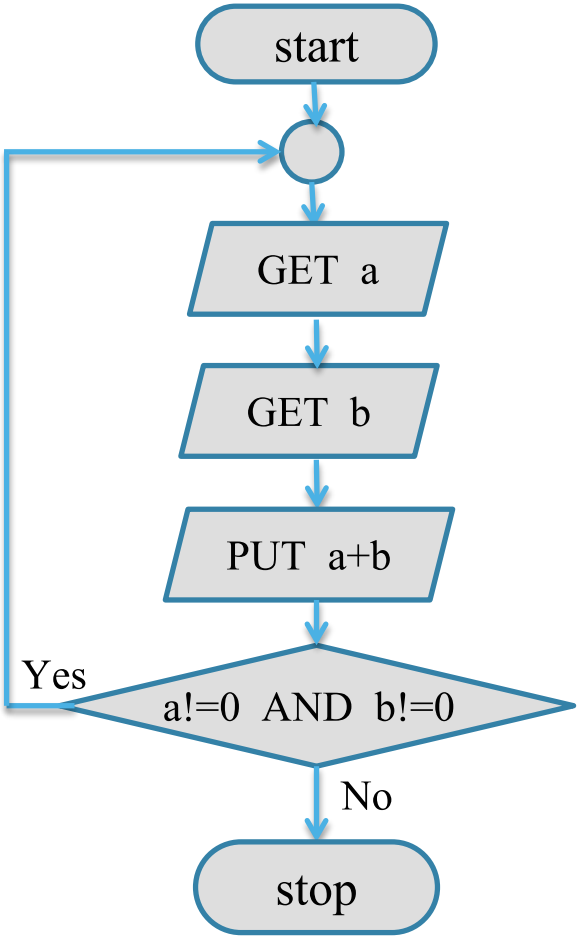



```
1  #include <stdio.h>
2
3  void main() {
4      int sum = 0, score;
5
6      printf("Enter first score (or 99 to quit)> ");
7      scanf("%d", &score);
8
9      while (score != 99) {
10         sum = sum + score;
11         printf("Enter next score (99 to quit)> ");
12         scanf("%d", &score);
13     }
14
15     printf("\nSum of exam scores is %d\n", sum);
16 }
```

Debug

ลูปควบคุมด้วยค่าเฝ้าดู

โปรแกรมบวกค่าตัวเลข 2 ตัว โดยรับค่าจนกว่าผู้ใช้จะป้อนค่าเป็น 0 ทั้ง 2 ตัว




```
#include <stdio.h>

void main() {
    int a,b;

    do {
        scanf("%d", &a);
        scanf("%d", &b);
        printf("%d\n", a+b);
    } while (a!=0 && b!=0);

}
```

การตรวจสอบความถูกต้องของ input

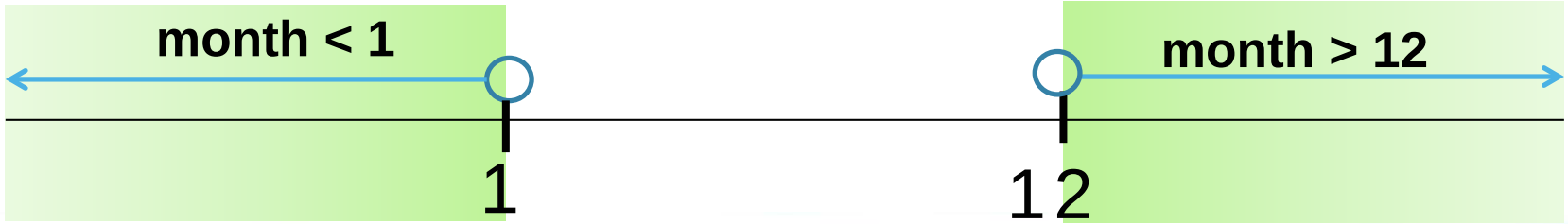
การใช้ if ตรวจสอบความถูกต้องไม่สามารถรับประกันได้ว่าผู้ใช้จะกรอกข้อมูลครั้งต่อไปได้ถูกต้องหรือไม่

```
1  #include <stdio.h>
2  void main() {
3      int LOW_MONTH = 1, HIGH_MONTH = 12;
4      int month;
5
6      printf("Enter birth month > ");
7      scanf("%d", &month);
8
9      if (month < LOW_MONTH || month > HIGH_MONTH) {
10         printf("Enter birth month > ");
11         scanf("%d", &month);
12     }
13     printf("\nYour birth month is %d\n", month);
14 }
```

การตรวจสอบความถูกต้องของ input

เปลี่ยนจากคำสั่ง if เป็น while เพื่อให้เกิดการตรวจสอบค่าที่กรอกในครั้งถัดไปได้

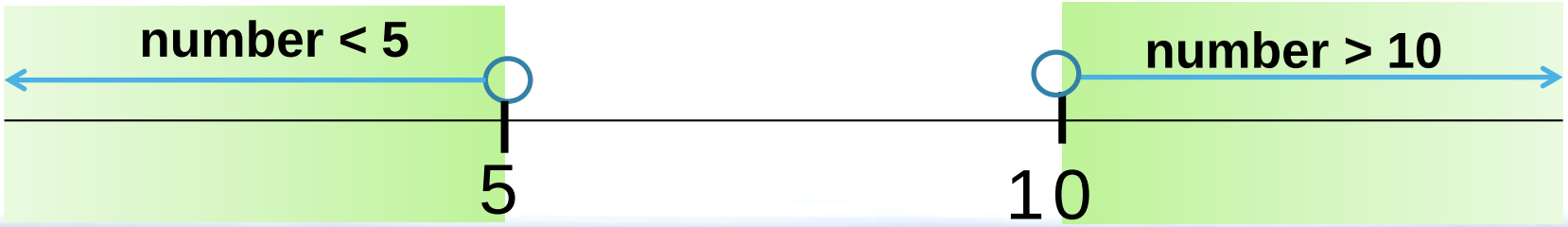
```
1  #include <stdio.h>
2  void main() {
3      int LOW_MONTH = 1, HIGH_MONTH = 12;
4      int month;
5
6      printf("Enter birth month > ");
7      scanf("%d", &month);
8
9      while (month < LOW_MONTH || month > HIGH_MONTH) {
10         printf("Enter birth month > ");
11         scanf("%d", &month);
12     }
13     printf("\nYour birth month is %d\n", month);
14 }
```



การตรวจสอบความถูกต้องของ input

โปรแกรมตรวจสอบว่า input ว่าอยู่ในช่วง 5 ถึง 10 หรือไม่ ถ้าไม่ใช่จะรับค่าเรื่อยๆ จนกว่าจะถูกต้อง

```
1  #include <stdio.h>
2
3  void main() {
4      int number;
5
6      do {
7          printf("Enter a number between 5 to 10> ");
8          scanf("%d", &number);
9      } while (number < 5 || number > 10);
10 }
```



การตรวจสอบความถูกต้องของ input

โปรแกรมตรวจสอบว่า input ว่าอยู่ในช่วง A ถึง E หรือไม่ ถ้าไม่ใช่จะรับค่าเรื่อยๆจนกว่าจะถูกต้อง

```
1  #include <stdio.h>
2
3  void main() {
4      char choice;
5
6      do {
7          printf("Enter a letter from A through E> ");
8          scanf(" %c", &choice);
9      } while (choice < 'A' || choice > 'E');
10 }
```

การตรวจสอบความถูกต้องของ input

โปรแกรมแสดงแม่สูตรคูณ โดยจะทำงานซ้ำหากผู้ใช้ตอบ Y แต่หากกดปุ่มอื่นๆ จะหยุดทำงาน

```
1  #include <stdio.h>
2  void main() {
3      char c;
4      int i,num;
5
6      do{
7          printf("\nEnter number = ");
8          scanf("%d",&num);
9
10         for(i=1; i<=12; i++)
11             printf("%d x %d = %d\n", num, i, num*i);
12
13         printf("Do you want to continue?(Y/N) > ");
14         scanf(" %c", &c);
15
16     } while(c=='Y' || c=='y');
17 }
```

กิจกรรม

โปรแกรมคำนวณค่านาทีเป็นชั่วโมงและนาทีที่เหลือ หากต้องการให้มีการทำงานไปเรื่อยๆ โดยถามผู้ใช้ว่าต้องการทำงานต่อหรือไม่ หากตอบ y จะทำงานอีกรอบ หากตอบ n จะหยุดทำงาน จงเพิ่มโค้ดโปรแกรมเพื่อให้เกิดการวนซ้ำดังกล่าว

```
#include <stdio.h>
void main() {
    int min, min2, hour;
    ……
    ……
        printf("Enter minutes: ");
        scanf("%d", &min);
        hour = min/60;
        min2 = min - (hour*60);
        printf("%d hour(s) %d minutes", hour, min2);
        ……
        ……
    ……
}
```

ตัวอย่างโจทย์ปัญหา

- จงเขียนโปรแกรมรับข้อมูลเงินเดือน และประสบการณ์การทำงาน แล้วคำนวณค่าโบนัสโดยมีเงื่อนไขดังนี้

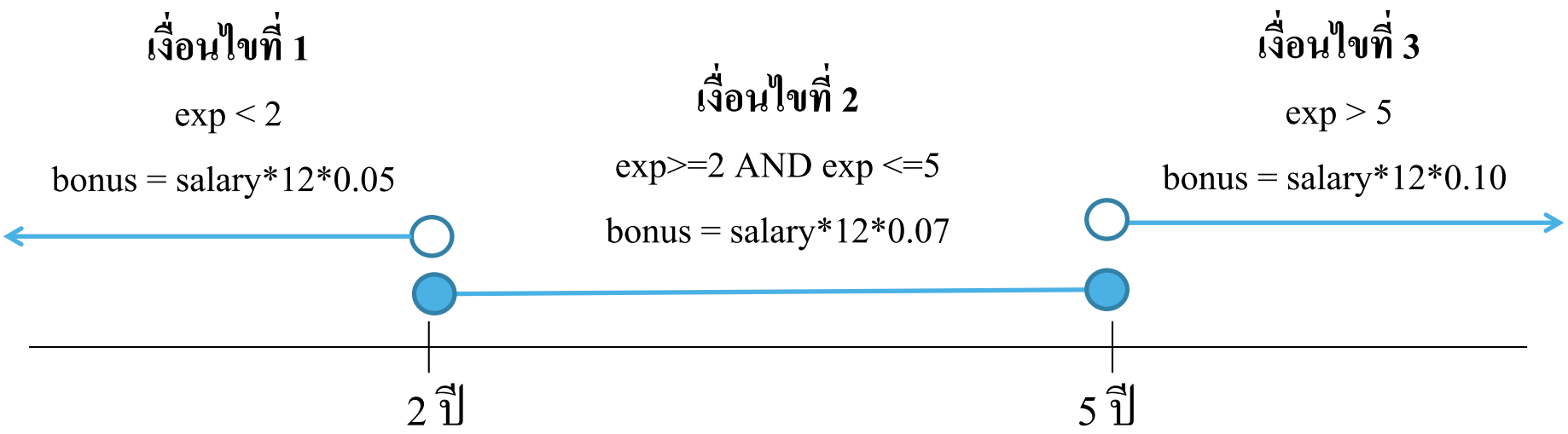
ประสบการณ์ (ปี)	โบนัสที่ได้ (บาท)
น้อยกว่า 2	5% ของเงินเดือนทั้งปี
2-5	7% ของ เงินเดือนทั้งปี
5 ขึ้นไป	10% ของเงินเดือนทั้งปี

- โปรแกรมจะวนลูปรับข้อมูลพนักงานได้หลายคน โดยถามผู้ใช้ว่าต้องการทำงานต่อหรือไม่ หากตอบ y จะทำงานอีกรอบ
- เมื่อผู้ใช้หยุดกรอกข้อมูลจะแสดงค่าเฉลี่ยโบนัสทั้งหมดที่กรอกเข้ามา

ขั้นตอนการวิเคราะห์โจทย์

- วิเคราะห์ส่วนที่ต้องทำงานใน 1 รอบการทำงาน
  - การตรวจสอบเงื่อนไข
  - การคำนวณโบนัสของพนักงานตามเงื่อนไขที่กำหนด
- วิเคราะห์ตัวแปรในการวนลูป
  - ตัวแปรควบคุมลูป + Sentinel Value
  - ตัวแปร Accumulator

1 รอบการทำงาน



1 รอบการทำงาน

```
printf("\nEnter salary : ");
scanf("%f",&salary);
printf("Enter experience : ");
scanf("%d",&exp);

if(exp < 2)
    bonus = salary * 12 * 0.05;
else if(exp>=2 && exp<=5)
    bonus = salary * 12 * 0.07;
else
    bonus = salary * 12 * 0.10;

printf("Bonus : %.2f\n", bonus);
```

คำสั่งทำซ้ำ (Loop)

```
int count = 0; // ตัวแปรสำหรับนับจำนวนรอบของลูป
float sum = 0; // ตัวแปรสะสมค่าผลรวม
char again = 'y'; // ตัวแปรเก็บค่าเฝ้าดู (กำหนดค่าเริ่มต้นเป็น y เพื่อให้เข้าลูปรอบแรกได้)

while (again!='n') {
    // โค้ดจากสไลด์หน้าที่แล้ว (1 รอบการทำงาน)
    printf("\nEnter salary : ");
    scanf("%f",&salary);
        . . .

    sum = sum + bonus;  // สะสมค่าจากตัวแปร bonus
    count++;  // นับจำนวนรอบของลูป
    printf("\nDo you want to calculate again? (y/n) > ");
    scanf(" %c", &again);
}

printf("\n\nBonus average is %.2f\n\n", sum/count);
```