การควบคุมทิศทางการทำงานของโปรแกรม

จากการเขียนโปรแกรมในบทเรียนที่ผ่านมา เป็นการเขียนโปรแกรมที่มีลักษณะโปรแกรมทำงานเป็นลำดับจากต้นจนจบโปรแกรม แต่ละคำสั่งจะทำงานเรียงกันไปตามลำดับที่มันปรากฏในโปรแกรม ซึ่งโปรแกรมในลักษณะนี้จะไม่มีความสามารถในการเลือกทำบางอย่างและไม่ทำบางอย่างได้ ไม่มีคำสั่งให้ทำกลุ่มของคำสั่งซ้ำ ไม่มีความสามารถในการเลือกทำกลุ่มของคำสั่ง ซึ่งโปรแกรมจริงๆจะต้องใช้คุณลักษณะที่กล่าวมานี้เป็นอย่างมาก โดยในโปรแกรมคอมพิวเตอร์ที่ซับซ้อนมักมีการสร้างทางเลือกสำหรับโปรแกรม เพื่อให้โปรแกรมทำงานได้ตามเงื่อนไขที่เกิดขึ้น

ดังนั้นเนื้อหาจะเกี่ยวกับการเขียนโปรแกรมเพื่อควบคุมทิศทางการทำงานของโปรแกรม

การควบคุมทิศทางการทำงานของโปรแกรม

- การควบคุมทิศทางแบบมีเงื่อนไข
  - ได้แก่ คำสั่ง if, if-else, if-else if และ switch
- การควบคุมทิศทางแบบวนรอบ
  - ได้แก่ คำสั่ง while, do-while และ for

การควบคุมทิศทางแบบมีเงื่อนไข

คำสั่ง if

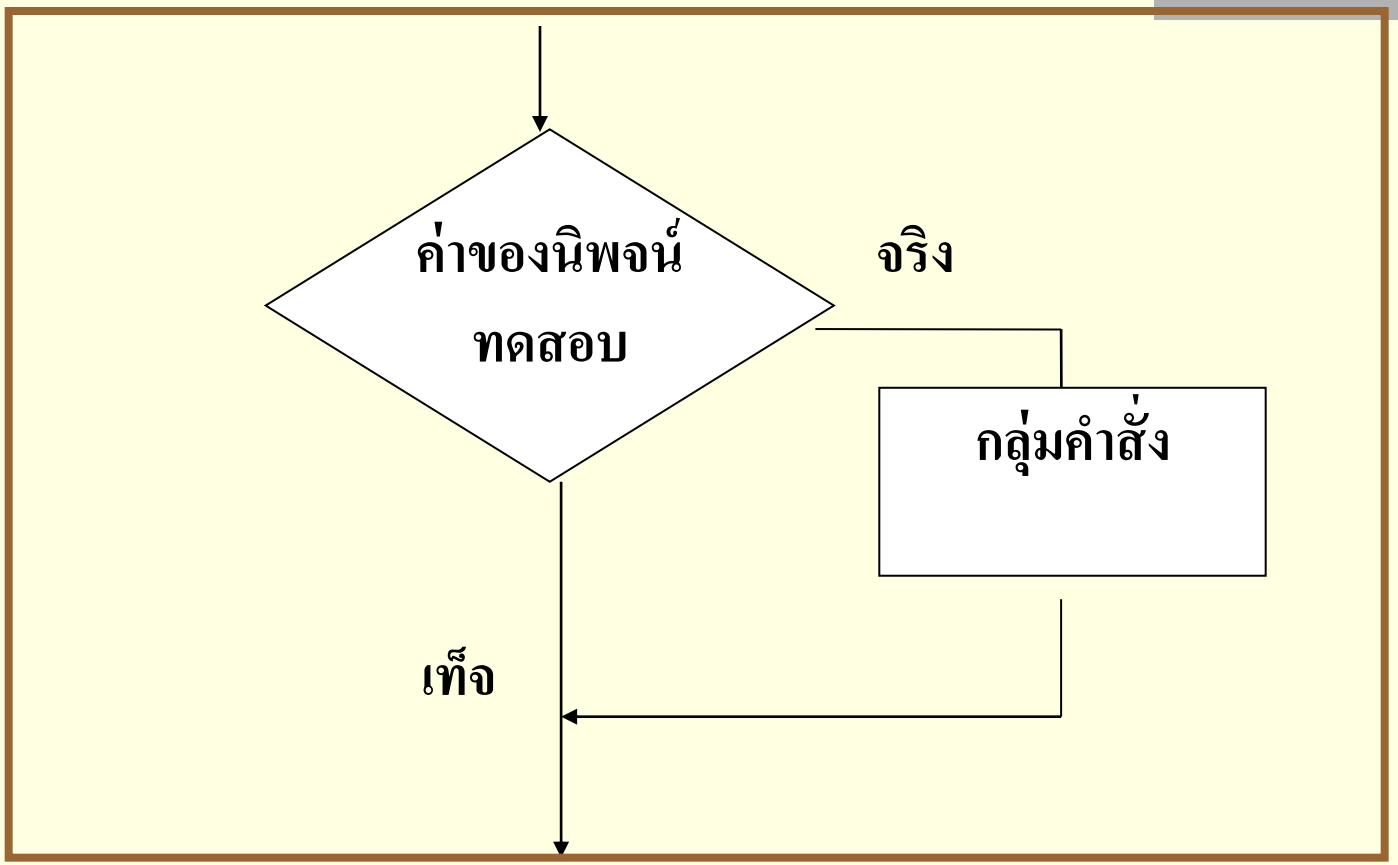
ในภาษา C คำสั่งสำหรับโครงสร้างการควบคุมเพื่อกำหนดทางเลือกคือประโยค if ซึ่งประกอบด้วยส่วนที่ทำหน้าที่ตรวจสอบเงื่อนไขที่เรียกว่า นิพจน์ทางตรรกศาสตร์ เราสามารถใช้โครงสร้างการควบคุมแบบ if ได้ในหลาย ๆ รูปแบบ

- คำสั่ง if เป็นคำสั่งที่ใช้เลือกการทำงานตามเงื่อนไข สามารถแบ่งตามลักษณะการทำงานได้ 3 แบบ ดังนี้
  1. คำสั่ง if แบบทางเดียว (if)
  2. คำสั่ง if แบบสองทาง (if-else)
  3. คำสั่ง if แบบหลายทาง (if-else if , nested- if)

1. ประโยคควบคุม if แบบ 1 ทางเลือก

เป็นประโยคควบคุม if ที่ง่ายที่สุด ในการตัดสินใจในประโยค if จะมีทางเลือกให้เพียงทางเดียวเท่านั้น โดยถ้าเงื่อนไขเป็นจริง ก็จะไปทำกลุ่มคำสั่งที่กำหนด แต่ถ้าเงื่อนไขเป็นเท็จก็จะหลุดออกจากประโยคควบคุม if

ผังงาน *if* statement แบบ 1 ทางเลือก



if (นิพจน์ทดสอบ) ประโยคที่ทำเมื่อผลจากนิพจน์ทดสอบเป็นจริง;

เช่น if (x<y) printf(“%d”,y);

คำสั่ง if

if (condition) statement;

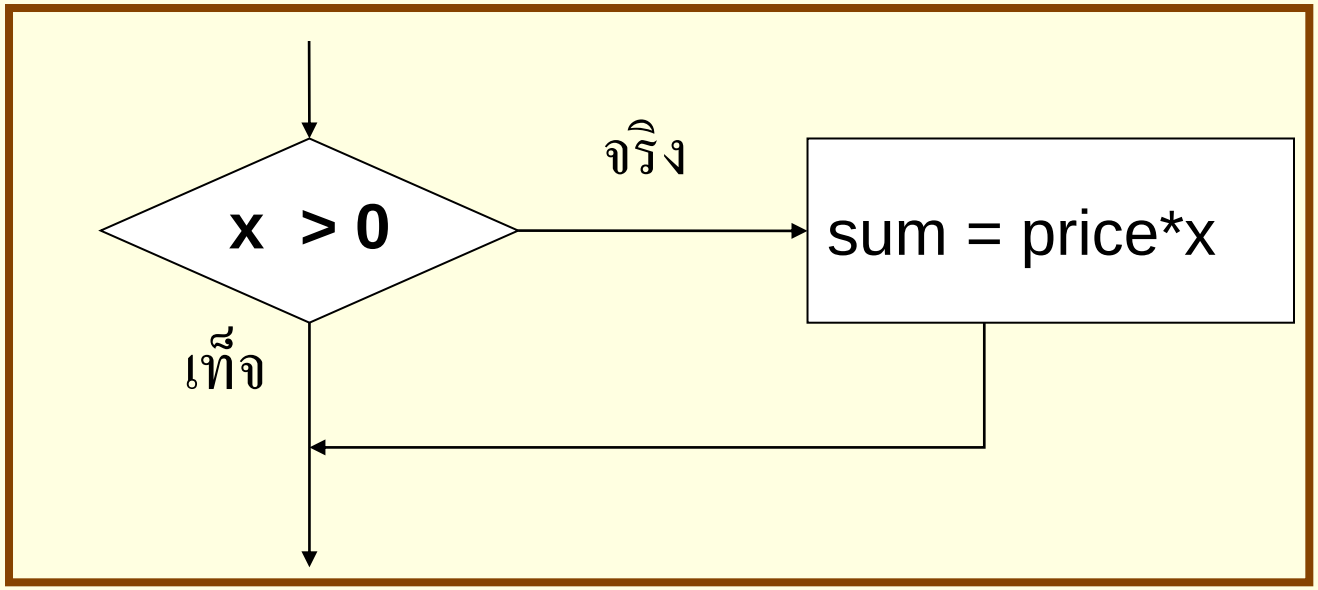
```
if  (condition)
{
        statement-1;
        statement-2;
        statement-3;
        …
        statement-n;
}
```

Condition : เงื่อนไขที่กำหนดขึ้นเพื่อใช้พิจารณาว่าจะทำหรือไม่ทำตามคำสั่ง โดยจะต้องเขียนไว้ภายในเครื่องหมาย ( ) ซึ่งเงื่อนไขอาจจะอยู่ในรูปของนิพจน์การคำนวณและเปรียบเทียบ หรือเป็นค่าของตัวแปรก็ได้

Statement : คำสั่งที่จะให้ทำงานถ้าผลการตรวจสอบเงื่อนไขเป็นจริง โดยอาจจะมีมากกว่า 1 คำสั่งก็ได้ แต่ต้องใช้เครื่องหมาย { } ครอบคำสั่งเหล่านั้นไว้ด้วย

ตัวอย่าง การใช้ประโยคควบคุม if แบบ 1 ทางเลือก

การตรวจสอบว่าจำนวนสินค้า(x) มากกว่าศูนย์หรือไม่ ถ้าใช่ ให้นำไปคูณราคาขายต่อหน่วย (price) แล้วเก็บไว้ในตัวแปร sum จะเขียนผังงานได้ดังนี้



จากตัวอย่างข้างต้นสามารถนำมาเขียนโปรแกรมภาษา C ได้ดังนี้

```
if (x>0)  sum = price*x;
```

ตัวอย่างโปรแกรมแบบ *if* statement

```
#include <stdio.h>
main()
{
      int  point;

      printf(“Enter your Exam Points : ”);
      scanf(“%d”,&point);
       if (point >= 50)
            printf(“You Passed.\n”);
}
```

ผลการรัน

Enter your Exam Points : 80
You Passed.

Enter your Exam Points : 25

ตัวอย่างโปรแกรมแบบ *if* statement

```
#include <stdio.h>
void main ( )
{
    int value;
    printf("Enter an integer: ");
    scanf("%d", &value);

     if (value = = 5) {
        printf("You entered a value of 5.\n");
        printf("This will now be changed to a 6!\n");
        value = value + 1;
      }
       printf("Goodbye\n");
}
```

ผลการรัน

```
Enter an integer: 75
Goodbye

Enter an integer: 5
You entered a value of 5.
This will now be changed to a 6!
Goodbye
```

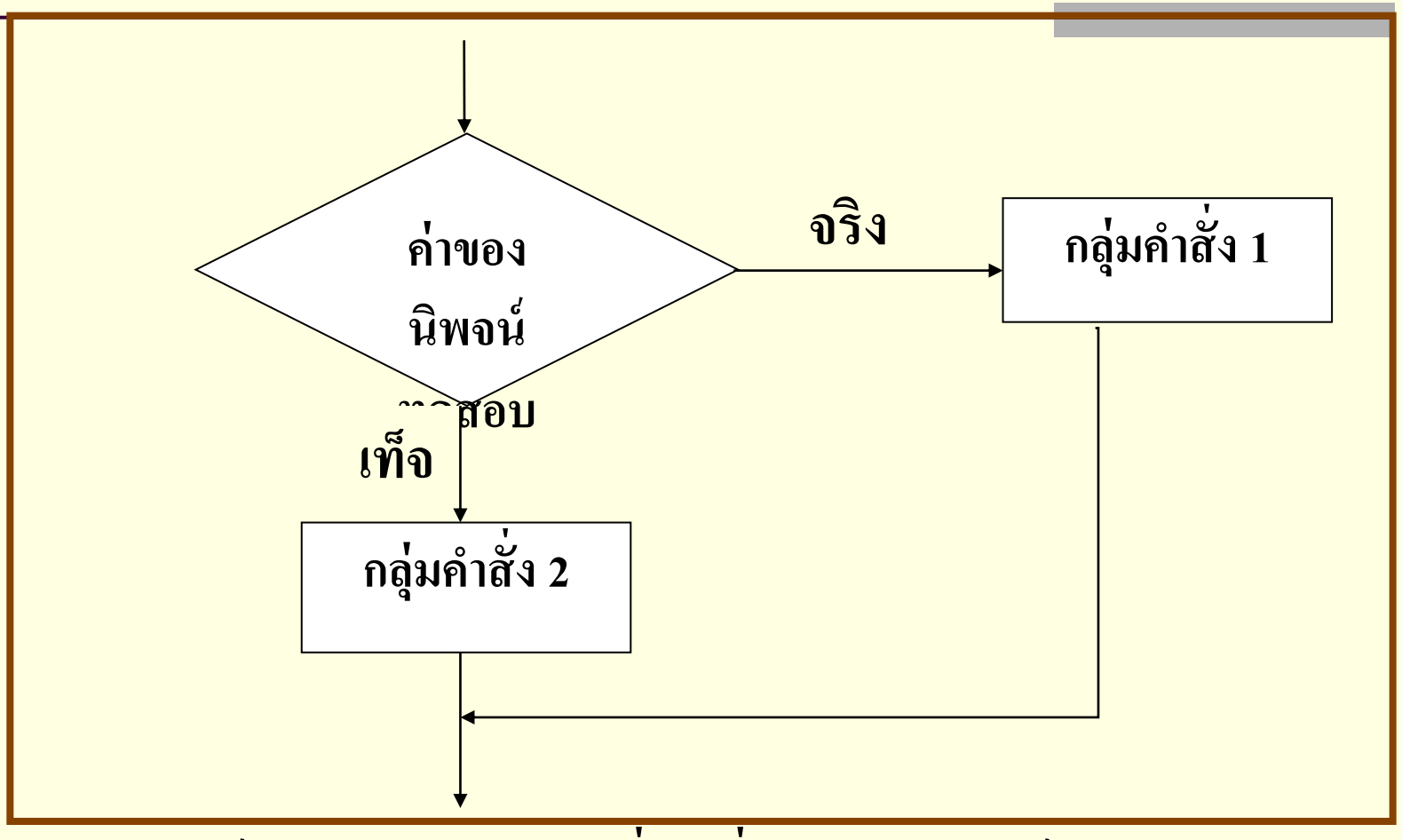
2. ประโยคควบคุม if แบบ 2 ทางเลือก

- คำสั่ง **if - else**

โครงสร้างแบบ if – else คล้ายกับประโยคควบคุม if แบบ 1 ทางเลือก เพียงแต่เพิ่มทางเลือกเข้ามา

ใช้ในกรณีที่เราต้องการให้มีทางเลือก 2 ทางให้ประมวลผล คือ ทางเลือกที่เงื่อนไขจริงมีค่าเป็นจริง และเงื่อนไขที่เป็นเท็จ

ผังงาน ประโยคควบคุม if แบบ 2 ทางเลือก



if (นิพจน์ทดสอบ) ประโยคที่ทำเมื่อผลจากนิพจน์ทดสอบเป็นจริง;

else ประโยคที่ทำเมื่อผลจากนิพจน์ทดสอบเป็นเท็จ;

คำสั่ง if – else

- คำสั่ง if – else จะใช้ในกรณีที่มีทางเลือกให้ทำงาน 2 ทางเลือก โดยมีรูปแบบการเรียกใช้งาน ดังนี้

```
if (condition)
        statement;
else
        statement;
```

- ถ้าผลการตรวจสอบเงื่อนไขเป็นจริง ก็ทำงานตามคำสั่งของ if
- แต่ถ้าผลการตรวจสอบเงื่อนไขเป็นเท็จ ก็ให้ทำงานตามคำสั่งของ else แทน

```
if  (condition)
{
        statement-1;
        statement-2;
        …
        statement-n;
}
else
{
        statement-1;
        statement-2;
        …
        statement-n;
}
```

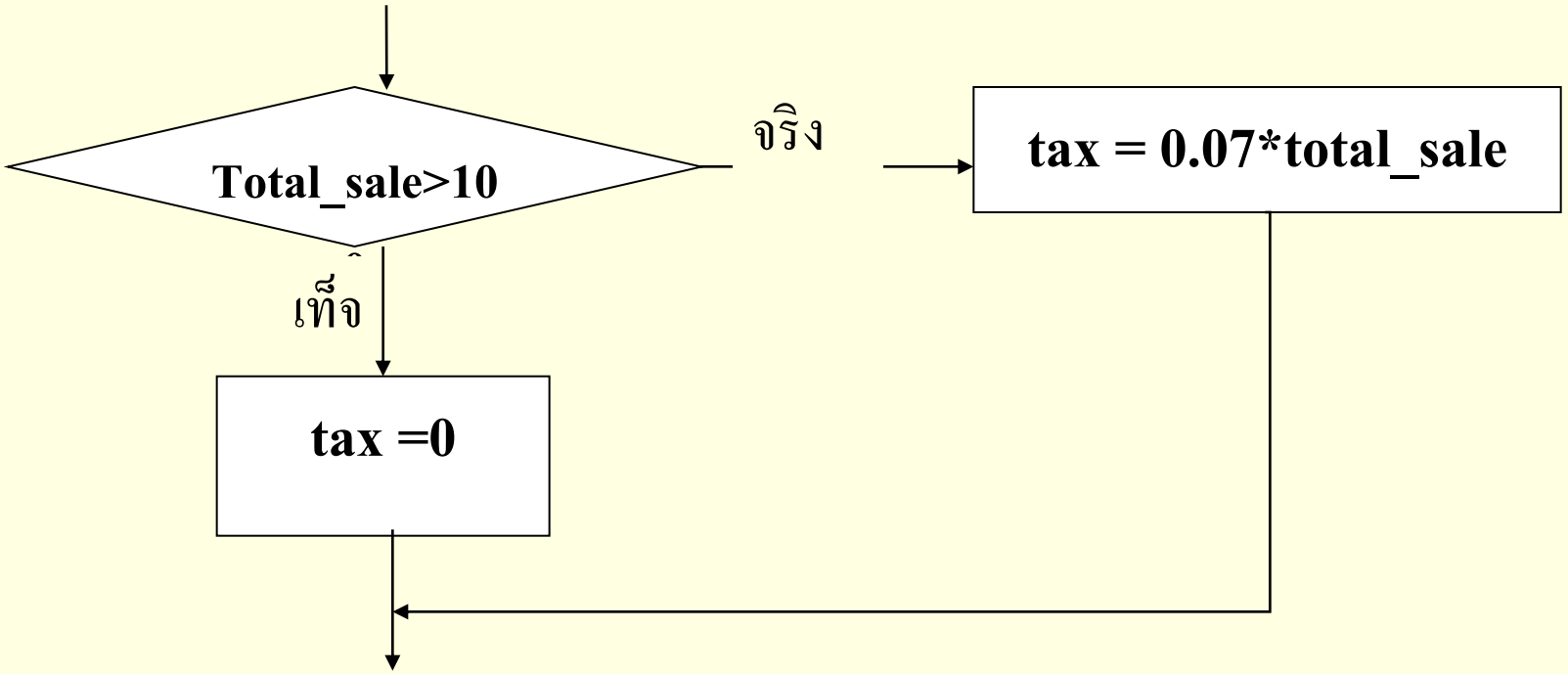
ตย. คำสั่ง if แบบสองทาง

- ถ้าค่าในตัวแปร **score** มากกว่าหรือเท่ากับ 50 ให้ตัวแปร **grade** มีค่า **'p'** แต่ถ้าค่าในตัวแปร **score** น้อยกว่า 50 ให้ตัวแปร **grade** มีค่า **'f'**

```
if  (score >= 50)
      grade =  'p';
else
      grade =  'f';
printf("score = %6.2f  grade = %c",score,grade);
```

ตัวอย่าง ประโยคควบคุม if แบบ 2 ทางเลือก

- การใช้ประโยคควบคุม if แบบ 2 ทางเลือก ตัวอย่างเช่น ในการตรวจสอบค่าตัวแปร totalsale ว่ามากกว่า 100 หรือไม่ ถ้ามากกว่าให้คิดภาษี 7% แต่ถ้าไม่ใช่ ก็ไม่ต้องคิดภาษี ดังนี้



- จากตัวอย่างผังงานข้างต้นสามารถเขียนคำสั่งในภาษา C ได้ดังนี้

```
if (total_sale>100)
        tax = 0.07 * total_sale;
else
        tax = 0;
```

ตัวอย่างโปรแกรมแบบ *if-else* statement

```
#include  <stdio.h>
main()
{
      int  point;

      printf("Enter your Exam Points : ");
      scanf("%d",&point);
       if (point >= 50)
            printf("You Passed.\n");
       else
            printf("You didn't Passed.\n");
}
```

ผลการรัน

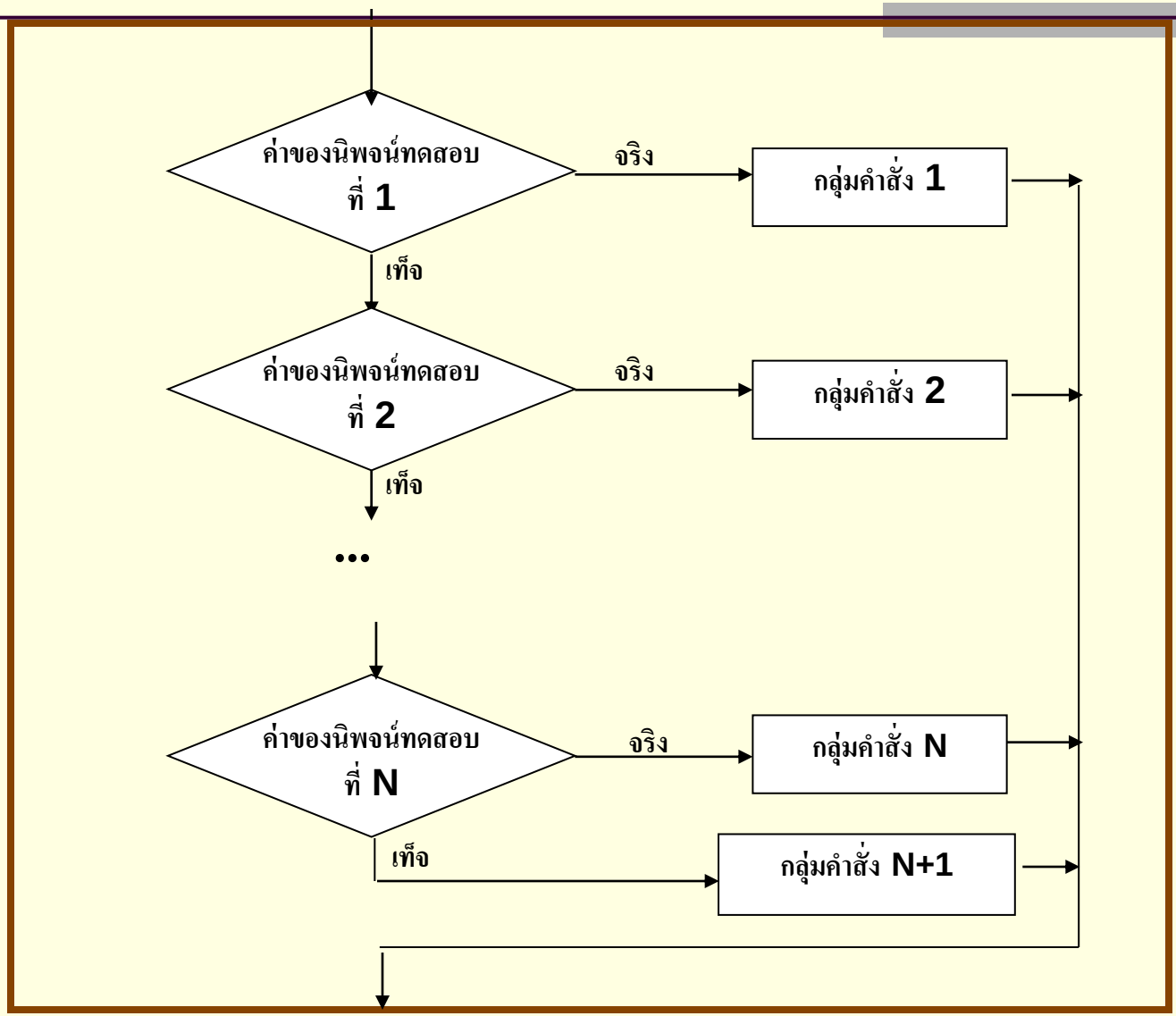
Enter your Exam Points : 80
You Passed.

Enter your Exam Points : 25
You didn't Passed.

3 ประโยคควบคุม if แบบหลายทางเลือก

- คำสั่ง **if-else if**
  - โครงสร้างแบบ if–else if เป็นประโยคควบคุม if ที่มีความซับซ้อนมากขึ้น จะใช้ในกรณีที่ทางเลือกมีมากกว่า 2 ทาง ตัวอย่างผังงานประโยคควบคุม if แบบหลายทางเลือก เป็นดังนี้

ผังงาน โครงสร้างการควบคุม if แบบหลายทางเลือก



```
if  (นิพจน์ทดสอบ 1)  { กลุ่มประโยคที่ทำเมื่อผลจากนิพจน์ทดสอบ 1 เป็นจริง }
else if  (นิพจน์ทดสอบ 2)  { กลุ่มประโยคที่ทำเมื่อผลจากนิพจน์ทดสอบ 2 เป็นจริง }
else if  (นิพจน์ทดสอบ 3)  { กลุ่มประโยคที่ทำเมื่อผลจากนิพจน์ทดสอบ 3 เป็นจริง }
                    .
                    .
                    .
else if  (นิพจน์ทดสอบ n)  { กลุ่มประโยคที่ทำเมื่อผลจากนิพจน์ทดสอบ n เป็นจริง }
else   { กลุ่มประโยคที่ทำเมื่อผลจากนิพจน์ทดสอบ n เป็นเท็จ }
          /* ทำประโยคนี้เมื่อไม่ได้ทำประโยคอื่นๆ */
```

คำสั่ง if– else if

- คำสั่ง if–else if จะใช้ในกรณีที่มีทางเลือกให้ทำงานมากกว่า 2 ทางเลือก โดยแต่ละทางเลือกมีเงื่อนไขต่างกัน ดังนั้นจึงต้องใช้คำสั่ง if หลายครั้งเพื่อกำหนดเงื่อนไขสำหรับแต่ละทางเลือก
  - โดยจะทำการตรวจสอบเงื่อนไขแรก ถ้าผลออกมาเป็นจริงก็จะทำงานตามคำสั่งของ if แต่ถ้าผลออกมาไม่จริงก็จะทำการตรวจสอบเงื่อนไขที่ 2 ซึ่งถ้าผลเป็นจริงก็จะทำงานตามคำสั่งของ else if นั้น ถ้าไม่จริงจะทำการตรวจสอบเงื่อนไขอื่นที่เรียงตามลำดับต่อไปจนเมื่อครบทุกเงื่อนไขแล้วถ้าผลยังคงไม่จริง ตัวแปลภาษา C จะทำงานตามคำสั่งที่กำหนดไว้ที่ else ซึ่งมีรูปแบบการเรียกใช้งาน ดังนี้

รูปแบบคำสั่ง if- else if

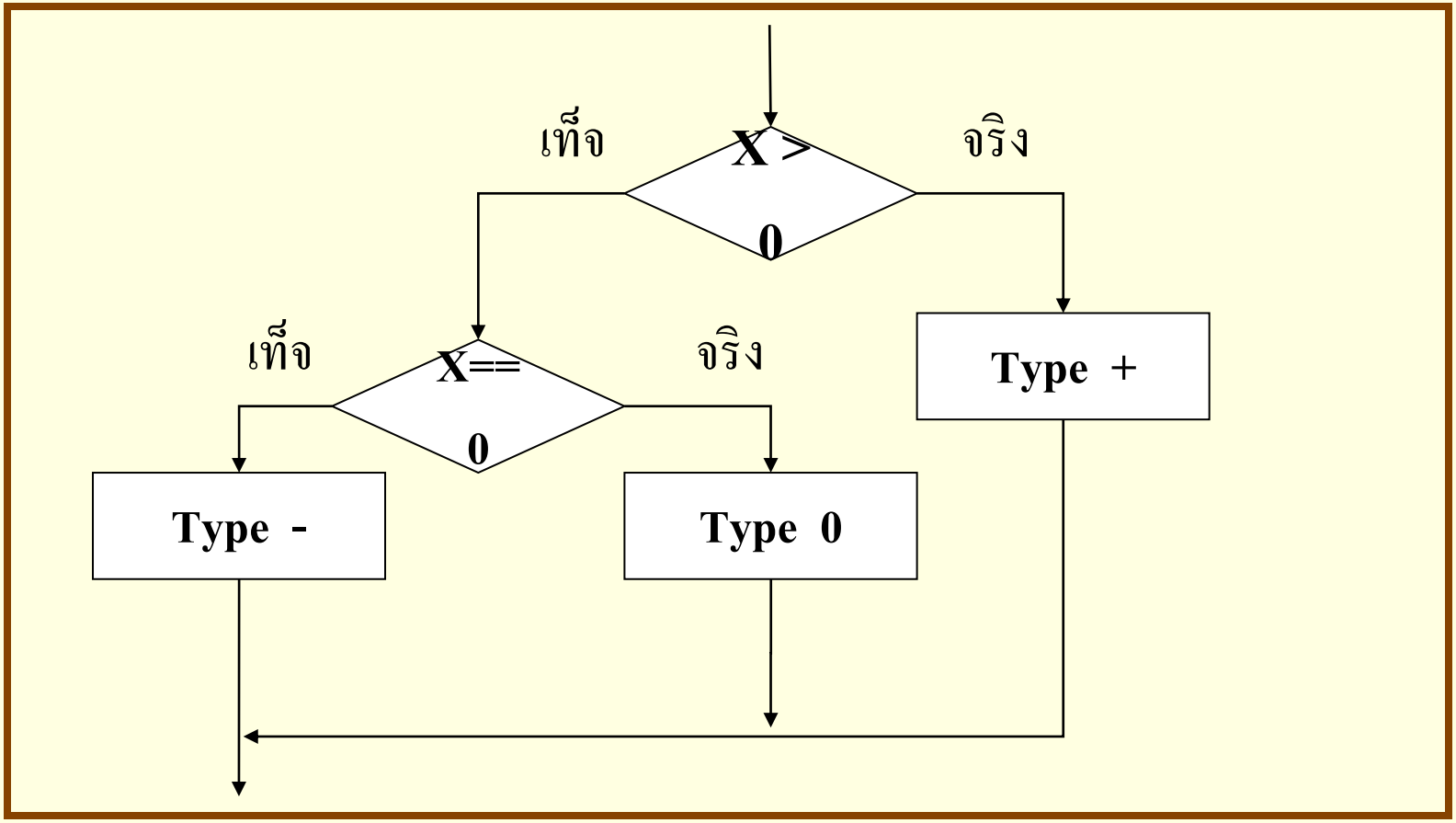
```
if (condition-1)
        statement;
else if (condition-2)
        statement;
else if (condition-3)
        statement;

            ...

else if (condition-n)
        statement;
else
        statement;
```

ตัวอย่าง การประยุกต์ใช้ทางเลือกหลายทาง

ในการตรวจสอบตัวเลขตัวหนึ่งว่าเป็นจำนวนเต็มบวก จำนวนเต็มลบ หรือจำนวนเต็มศูนย์ สามารถเขียนผังงานได้ดังนี้



- จากตัวอย่างผังงานข้างต้นสามารถเขียนคำสั่งในภาษา C ได้ดังนี้

```
if (x>0)                          /* เงื่อนไขที่ 1 */
    printf("Type Positive Number\n");      /* ทางเลือกที่ 1 */
else if (x==0)                    /* เงื่อนไขที่ 2 */
    printf("Type Zero Number\n");          /* ทางเลือกที่ 2 */
else
    printf("Type Negative Number\n");      /* ทางเลือกที่ 3 */
```

ตัวอย่างโปรแกรมแบบ *if-else if* statement

```
main()
{
      int  point;
      printf(“Enter your Exam Points : ”);
      scanf(“%d”,&point);
       if  ((point<= 100)&&(point>=80))
                  printf(“Grade A\n”);
       else if ((point< 80)&&(point>=70))
                  printf(“Grade B\n”);
       else if ((point< 70)&&(point>=60))
                  printf(“Grade C\n”);
       else if ((point< 60)&&(point>=50))
                  printf(“Grade D\n”);
       else
                  printf(“Grade F\n”);
}
```

คำสั่ง if ซ้อน if (nested if)

- คำสั่ง if ซ้อน if จะใช้กับปัญหาที่มีเงื่อนไขซับซ้อน อย่างเช่น กำหนดว่าต้องเป็นจำนวนคู่และมีค่าไม่เกิน 20 หรือ เงื่อนไขต้องเป็นเพศหญิงอายุไม่เกิน 30 ปี เป็นต้น
- ซึ่งคำสั่ง if ซ้อน if ก็คือคำสั่ง if ตามปกติ แต่เรานำมาเขียนซ้อนกันมากกว่า 1 ชั้น ซึ่งมีรูปแบบการเรียกใช้งาน ดังนี้

```
if (condition-1)
{
        if (condition-2)
        {
                ...
                if (condition-n)
           statement;
         }
}
```

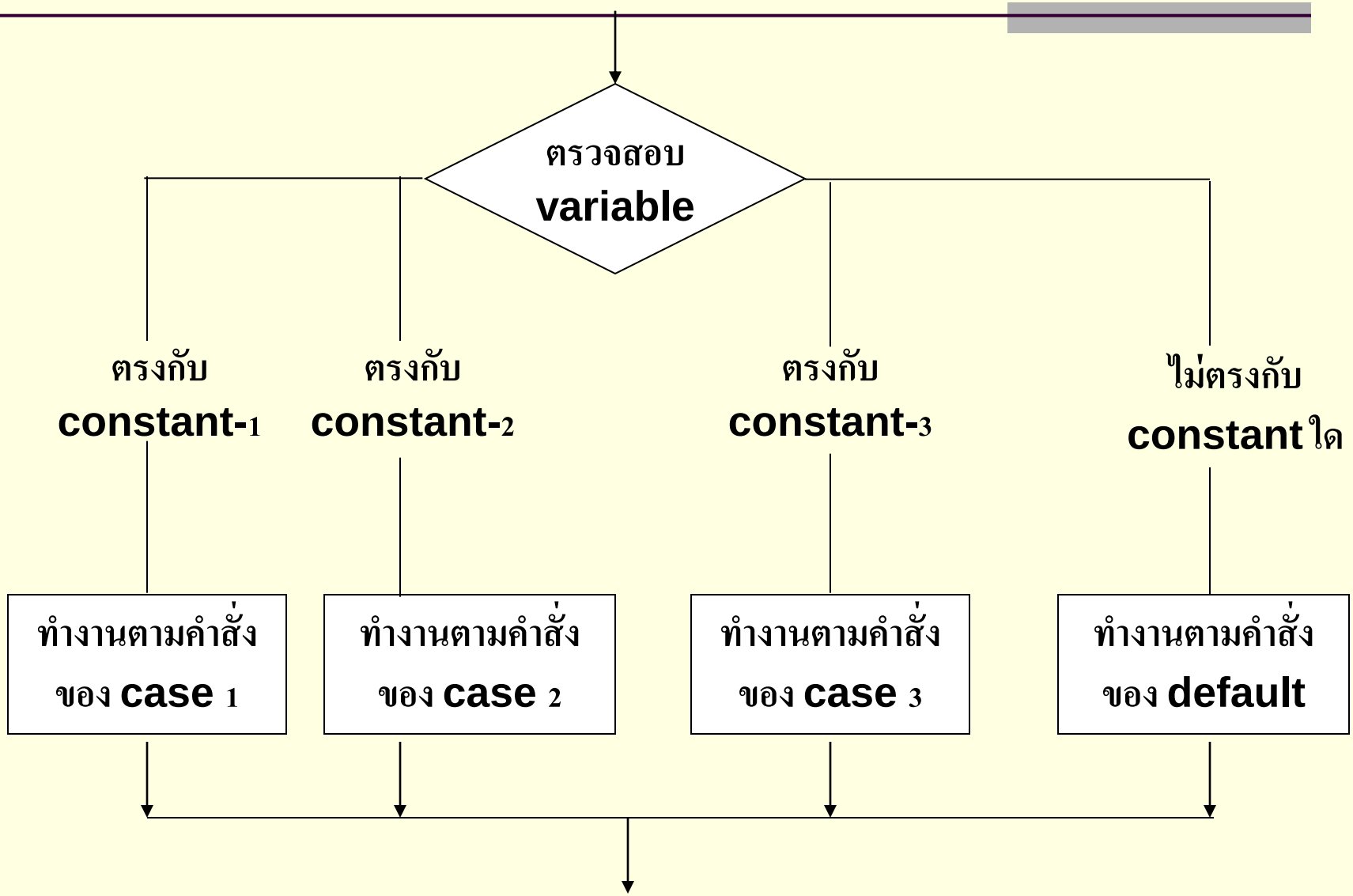
ตัวอย่างโปรแกรมแบบ *nested-if* statement

```
main()
{
      int  num;
      printf(“Enter your number : ”);
      scanf(“%d”,&num);
      if  (num>=20)
       {
                if (num<=30)
                {
                          if (num%2==1)
                                printf(“Right number\n”);
                          else
                                printf(“Wrong number\n”);
                 }
                 else
                 printf(“Large number\n”);
       }
       else
       printf(“Small number\n”);
}
```

4. ประโยคควบคุม switch

ในการเขียนโปรแกรมภาษา C มีประโยคควบคุม switch ที่ช่วยอำนวยความสะดวกในการเขียนโปรแกรมเพื่อสร้างทางเลือกหลายทาง นอกเหนือจากกรณีการใช้ if ดังกล่าวมาแล้ว ประโยคควบคุม switch มีลักษณะการทดสอบเงื่อนไขกับค่าคงที่ เพื่อสร้างทางเลือก โดยจะทำการทดสอบทีละค่า

ผังงาน สำหรับประโยคควบคุม switch



คำสั่ง switch

- คำสั่ง switch จะใช้ในกรณีที่มีทางเลือกให้ทำงานหลายทางเลือกโดยใช้เงื่อนไขร่วมกัน ซึ่งตัวแปรภาษา C จะทำการตรวจสอบเงื่อนไขเพียงครั้งเดียว ผลจากการตรวจสอบเงื่อนไขจะถูกนำไปพิจารณาอีกครั้งว่าจะทำงานตามทางเลือกใด โดยมีรูปแบบการเรียกใช้งาน ดังนี้

รูปแบบการใช้คำสั่ง switch

```
switch(variable) {
   case  constant-1 :
         statement sequence;
         break;
   case  constant-2 :
         statement sequence;
         break;
         .
         .
         .
   case  constant-n :
         statement sequence;
         break;
   default :
         statement sequence;
}
```

- **variable :** ตัวแปรที่ทำหน้าที่ควบคุมการทำงานของคำสั่ง **switch** ซึ่งจะเป็นตัวแปรชนิด **int, char** หรือ นิพจน์ก็ได้ โดยค่าของตัวแปรนี้ถ้าตรงกับ **constant** ที่อยู่ใน **case** ใดก็จะทำคำสั่งที่อยู่ใน **case** นั้นนั่นเอง
- **constant-1, constant-2, ...constant-n :** เป็นเงื่อนไขที่จะให้เลือกไปทำงานตามที่ต้องการ โดยจะต้องมีค่าเป็นค่าคงที่ ชนิด **int, char** เท่านั้น
- **statement sequence :** กลุ่มของคำสั่งที่ต้องการให้ทำงานตามเงื่อนไขแต่ละ **case**
- **break :** เป็นคำสั่งสำหรับออกจากการทำงานของ **switch** โดยให้ออกจากบล็อกของ **case** แล้วไปทำคำสั่งที่อยู่ต่อจาก **switch**
- **default :** ในกรณีที่ค่าของตัวแปร **variable** ไม่ตรงกับค่าของ **constant** ใน **case** ใดเลย ตัวแปรภาษา C ก็จะมาทำงานตามคำสั่งใน **default** ก่อนที่จะออกไปทำงานตามคำสั่งอื่นที่อยู่ต่อจาก **switch**

ตัวอย่างโปรแกรมแบบ *switch* statement

```
main()
{  char  grade;
     printf("Enter your grade : ");
     scanf("%c",&grade);
     switch (grade) {
          case 'A':
                printf("100 - 80 points\n");
                break;
          case 'B':
                printf("79 - 70 points\n");
                break;
          case 'C':
                printf("69 - 60 points\n");
                break;
          case 'D':
                printf("59 - 50 points\n");
                break;
          default :
                printf("49 - 0 points\n");
          }
}
```

ผลการรัน

```
Enter your grade : C
69 - 60 points
```

กรณีที่ไม่มีคำสั่ง break; ต่อท้าย

```
main()
{  char  grade;
     printf(“Enter your grade : ”);
     scanf(“%c”,&grade);
     switch (grade) {
          case ‘A’:
               printf(“100 - 80 points\n”);
          case ‘B’:
               printf(“79 - 70 points\n”);
          case ‘C’:
               printf(“69 - 60 points\n”);
          case ‘D’:
               printf(“59 - 50 points\n”);
          default :
               printf(“49 - 0 points\n”);
          }
}
```

ผลการรัน

```
Enter your grade : C
69 - 60 points
59 - 50 points
49 - 0 points
```

คำสั่ง GOTO

- เป็นการสั่งให้ข้ามไปทำงานยังตำแหน่งที่มีชื่อ (label) ตามที่ระบุโดยไม่มีเงื่อนไข
- รูปแบบ

```
goto  statement-label;
```

- **statement-label** หมายถึง ชื่อประจำคำสั่ง (labal) ที่ต้องการให้ไปทำงาน

ตัวอย่าง

```
.............
.............
goto loop1;
.............
.............
```

- หมายความว่าให้ไปทำคำสั่งที่มีชื่อประจำคำสั่งเป็น loop1 ดังนี้

```
loop1 :.............
        .............
        .............
        .............
        goto loop1;
      .............
      .............
```

ตัวอย่าง โปรแกรมที่ใช้คำสั่ง *goto*

```
#include <stdio.h>
main( )
{   float  score;
    again:
            printf("\nEnter score  : ");
            scanf("%f", &score);
            if  (score <0 || score > 100 )
          goto again;
            printf("your  score is  %6.2f", score);
}
```