บทที่ 5
โครงสร้างควบคุมแบบเลือก
(Selection control structure)

โครงสร้างการควบคุม จะเป็นการควบคุมทิศทางการกระทำการ (execution) ในโปรแกรมหรือฟังก์ชัน ซึ่งคำสั่งในภาษา C ที่ใช้ในการควบคุมการทำงานของโปรแกรม จะแบ่งออกเป็น 3 แบบ คือ

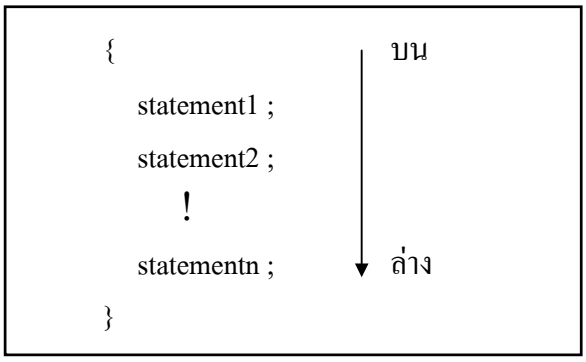
5.1 แบบตามลำดับ (sequence)

5.2 แบบเลือก (selection)

5.3 แบบทำซ้ำ (repetition)

5.1 โครงสร้างควบคุมแบบตามลำดับ (sequence)

จะมีลักษณะการกระทำการ (execute) จากข้อความสั่งแรกไปตามลำดับ จากบนลงล่างจนถึงข้อความสั่งสุดท้าย ดังรูปที่ 5.1



รูปที่ 5.1

5.2 โครงสร้างควบคุมแบบเลือก

เป็นโครงสร้างที่ใช้ในการเลือกทำคำสั่งในทิศทางใดนั้นขึ้นอยู่กับเงื่อนไขที่ถูกทดสอบ โดยข้อความสั่งที่ใช้ในการเลือกในภาษา C จะมีอยู่ 2 แบบ คือ

5.2.1 ข้อความสั่ง if (if statement)

5.2.2 ข้อความสั่ง switch (switch statement)

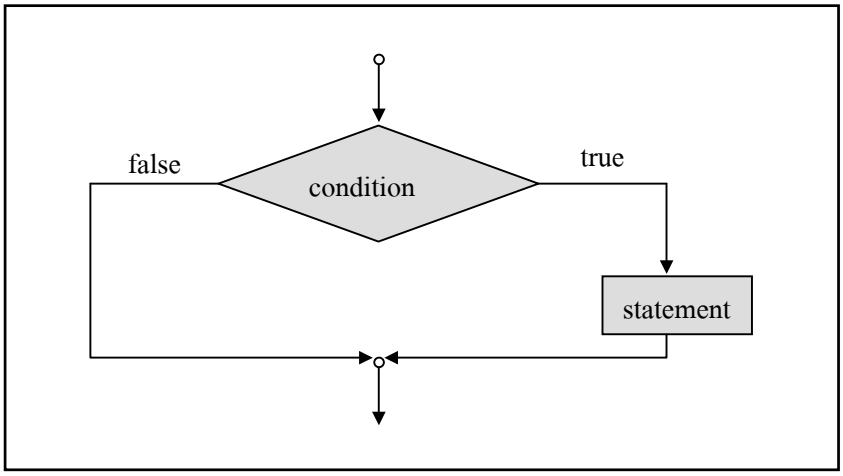
5.2.1 ข้อความสั่ง if

5.2.1.1 รูปแบบทั่วไปของข้อความสั่ง if แบบ มี 1 ทางเลือก แสดงได้ดังรูปที่ 5.2

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

รูปที่ 5.2

โดยถ้า condition เป็นจริง (ตัวเลขไม่ใช่ 0) แล้ว statement จะถูกกระทำการ (execute) แต่ถ้า condition เป็นเท็จ (ตัวเลขเป็น 0) แล้ว statement จะถูกข้าม ซึ่งสามารถแสดงเป็นผังงานได้ดังรูปที่ 5.3



รูปที่ 5.3

โดย statement ในที่นี้อาจจะเป็น 1 ข้อความสั่ง หรือหลายข้อความสั่งก็ได้ ถ้ามีมากกว่า 1 ข้อความสั่ง ต้องถูกปิดล้อมด้วยเครื่องหมาย { และ } แสดงได้ดังรูปที่ 5.4

```
if (condition)
    {
        statement1 ;
        statement2 ;
            !
        statementn ;
    }
```

รูปที่ 5.4

หมายเหตุ ในส่วนของ condition จะต้องถูกปิดล้อมเครื่องหมาย (และ)

ตัวอย่างโปรแกรมที่ 5.1 เป็นโปรแกรมที่มีเพียง 1 ข้อความสั่ง เมื่อเงื่อนไขที่ถูกทดสอบเป็นจริง

```
#include <stdio.h>
main ()
{
    float number ;  /* Value supplied by user. */

    printf ( “Give me a number from 1 to 10 = > ” ) ;
    scanf ( “%f”, &number ) ;
    if (number > 5)
        printf ( “Your number is larger than 5. \n” ) ;  <-------- 1 ข้อความสั่ง
    printf ( “%f was the number you entered.”, number ) ;
}
```

โดยใช้โปรแกรมนี้จะให้ผู้ใช้ป้อนตัวเลขจำนวนเต็ม 1 ถึง 10 ถ้าป้อนตัวเลข 1 ถึง 5 ในที่นี้สมมติป้อนเลข 3 โปรแกรมก็จะแสดงผลที่จอภาพ ดังนี้

```
3 was the number you entered.
```

ถ้าผู้ใช้ป้อนตัวเลขจำนวนเต็มมากกว่า 5 สมมติป้อนเลข 7 โปรแกรมก็จะแสดงผลที่จอภาพ ดังนี้

```
Your number is larger than 5.

7 was the number you entered.
```

เนื่องจากผลลัพธ์ที่ได้ เกิดจากข้อความสั่ง if ดังนี้

```
if (number > 5)
    printf ( “Your number is larger than 5. \n” ) ;
```

ถ้าค่าที่ป้อนเข้ามามากกว่า 5 จะทำให้นิพจน์ความสัมพันธ์ของ number > 5 เป็นจริง

ข้อความสั่ง printf (“Your number is larger than 5. \n”) ; นี้จะถูกกระทำการ

ถ้าเป็นเท็จข้อความสั่ง printf ด้านบนนี้ จะไม่ถูกกระทำการ

ตัวอย่างโปรแกรมที่ 5.2 เป็นโปรแกรมที่แสดงค่าสัมบูรณ์ของจำนวนเต็ม

```
/* Calculate the absolute value of an integer */
#include <stdio.h>
main ()
{
    int number ;

    printf ( “Type in your number : ” ) ;
    scanf ( “%d”, &number ) ;

    if ( number < 0 )
        number =  –number ;

    printf ( “The absolute value is %d\n”, number ) ;
}
```

```
Type in your number : –100
The absolute value is 100
```

```
Type in your number : 2000
The absolute value is 2000
```

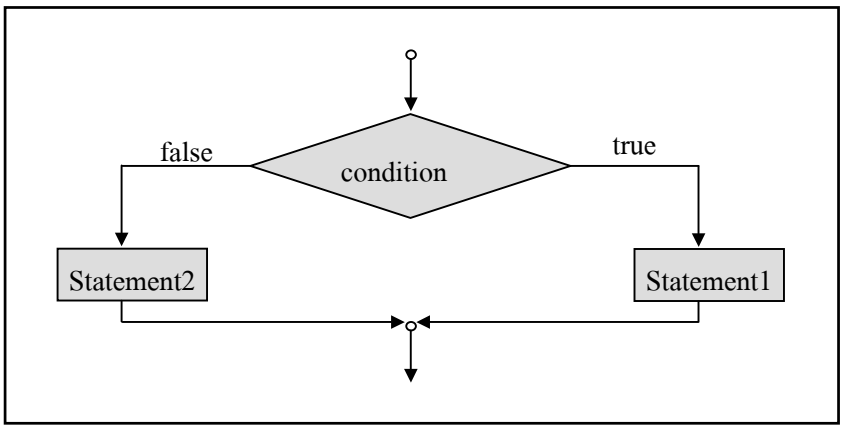
5.2.1.2 รูปแบบทั่วไปของข้อความสั่ง if แบบมี 2 ทางเลือก แสดงได้ดังรูปที่ 5.5

```
if (condition)
    statement1 ;
else
    statement2 ;
```

รูปที่ 5.5

โดยถ้า condition มีค่าเป็นจริง (ค่าไม่ใช่ 0) แล้ว statement1 จะถูกกระทำการ และ statement2 จะถูกข้าม และ

ถ้า condition มีค่าเป็นเท็จ (ค่าเป็น 0) แล้ว statement2 จะถูกกระทำการ และ statement1 จะถูกข้าม ซึ่งสามารถแสดงเป็นผังงานได้ดังรูปที่ 5.6



รูปที่ 5.6

ในทำนองเดียวกัน statement1, statement2 อาจจะเป็น 1 ข้อความสั่ง หรือหลายข้อความสั่งก็ได้ โดยถ้ามีมากกว่า 1 ข้อความสั่ง ต้องถูกปิดล้อมด้วยเครื่องหมาย { และ } แสดงได้ดังรูปที่ 5.7

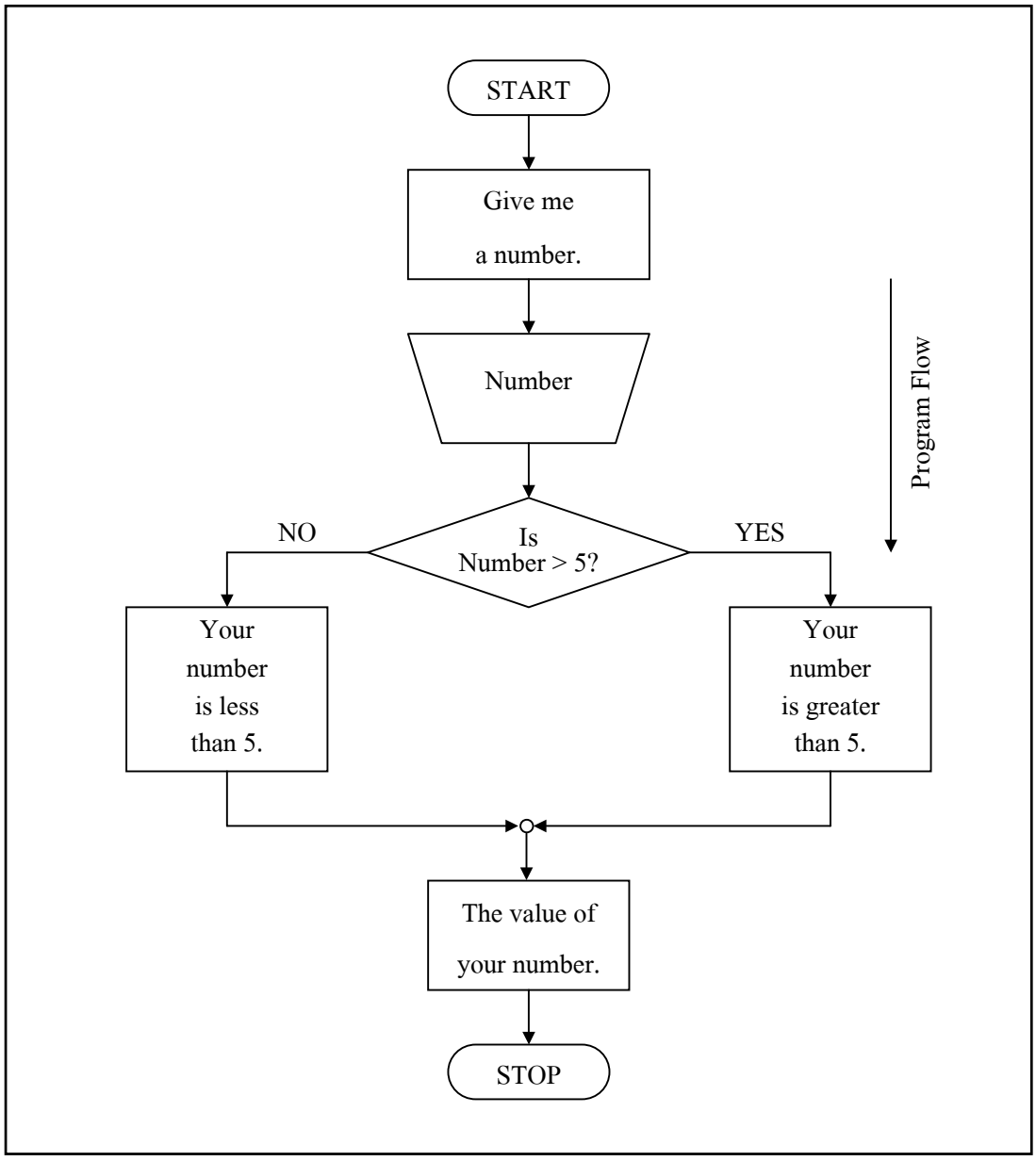
```
if (condition)
   {  statement1 ;
        !
      statementn ;
   }
else
   {  statement1 ;
        !
      statementn ;
   }
```

รูปที่ 5.7

ตัวอย่างโปรแกรมที่ 5.3 เป็นโปรแกรมแสดงการรับค่าจำนวนเข้ามา แล้วนำมาตรวจสอบกับเงื่อนไข number > 5.0

```
#include <stdio.h>
main ()
{
    float number ;       /* User number value. */

    printf ( "\n\nGive me a number from 1 to 10 = >" ) ;
    scanf ( "%f", &number ) ;

    if (number > 5.0)
        printf ( "Your number is greater than 5. \n" ) ;
    else
        printf ( "Your number is less than or equal to 5. \n" ) ;
    printf ( "The value of your number was %f", number ) ;
}
```

ถ้าป้อนตัวเลขที่มากกว่า 5 จะแสดงข้อความ Your number is greater than 5. บนจอภาพ ถ้าป้อนตัวเลขที่น้อยกว่าหรือเท่ากับ 5 จะแสดงข้อความ Your number is less than or equal to 5. บนจอภาพ และโปรแกรมนี้สามารถแสดงผังงานได้ดังรูปที่ 5.8

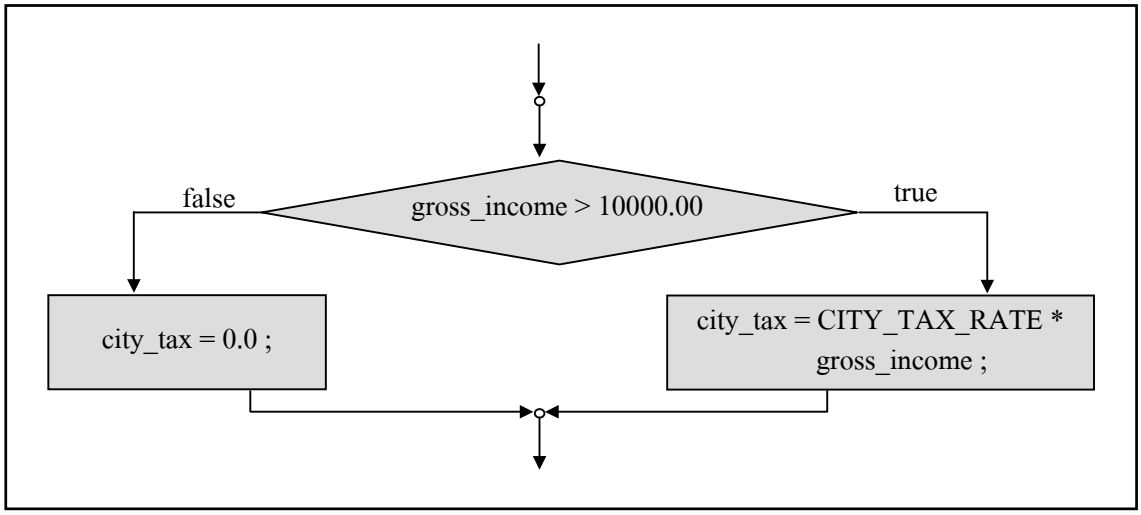


รูปที่ 5.8

ตัวอย่างโปรแกรมที่ 5.4 เป็นโปรแกรมแสดงการคำนวณภาษีเงินได้

```
#include <stdio.h>

const double CITY_TAX_RATE =0.0175 ;

int main (void)   {
    /* Variable declarations : */

    double gross_income ;
    double city_tax ;

    /* Function body : */

    printf ( “A PROGRAM THAT COMPUTERS CITY INCOME TAX \n” ) ;
    printf ( “Enter gross income : ” ) ;
    scanf ( “%1f”, &gross_income ) ;

    if (gross_income > 10000.00)
        city_tax = CITY_TAX_RATE * gross_income ;
    else
        city_tax = 0.0 ;
    /* end if */
    printf ( “City tax is %f dollars.”, city_tax ) ;
    return 0 ;
}   /* end function main */
```

และโปรแกรมนี้จะให้ผู้ใช้ป้อนข้อมูลเข้ามา และสามารถแสดงผังงานได้ดังรูปที่ 5.9

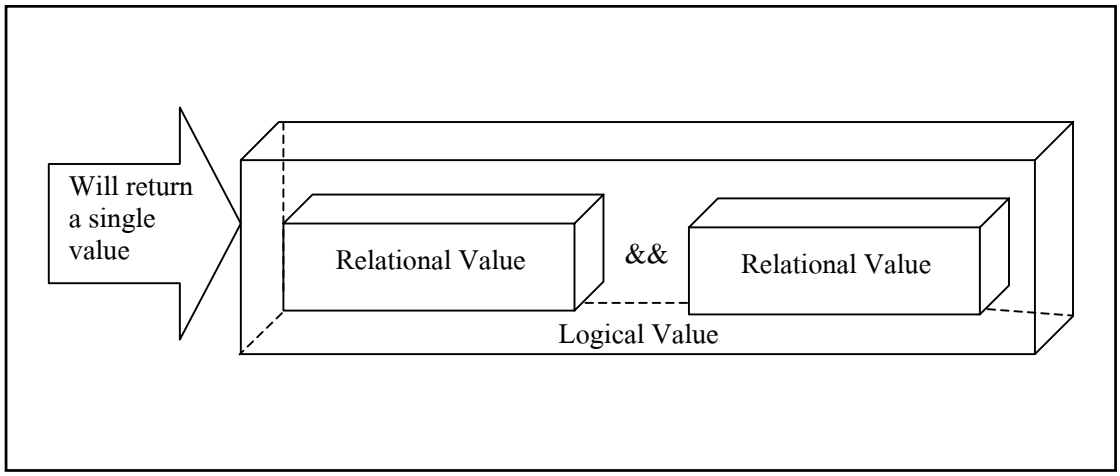


รูปที่ 5.9

ตัวอย่างโปรแกรมที่ 5.5 เป็นโปรแกรมที่มีการนำตัวดำเนินการ AND มาใช้

```
#include <stdio.h>

main ()
{
    float number ;    /* User input number. */

    printf ( "\n\nGive me a number from 1 to 100 = > " ) ;
    scanf ( "%f", &number ) ;
                                          ตัวดำเนินการความสัมพันธ์ AND
    if ( (number > = 1.0) && (number < = 10.0) )
        printf ( "You gave me a number between 1 and 10." ) ;
}
```

โปรแกรมนี้ ถ้าผู้ใช้ป้อนตัวเลขตั้งแต่ 1.0 ถึง 10.0 จะทำให้เงื่อนไขเป็นจริง ซึ่งเงื่อนไขสามารถแสดงได้ดังรูปที่ 5.10

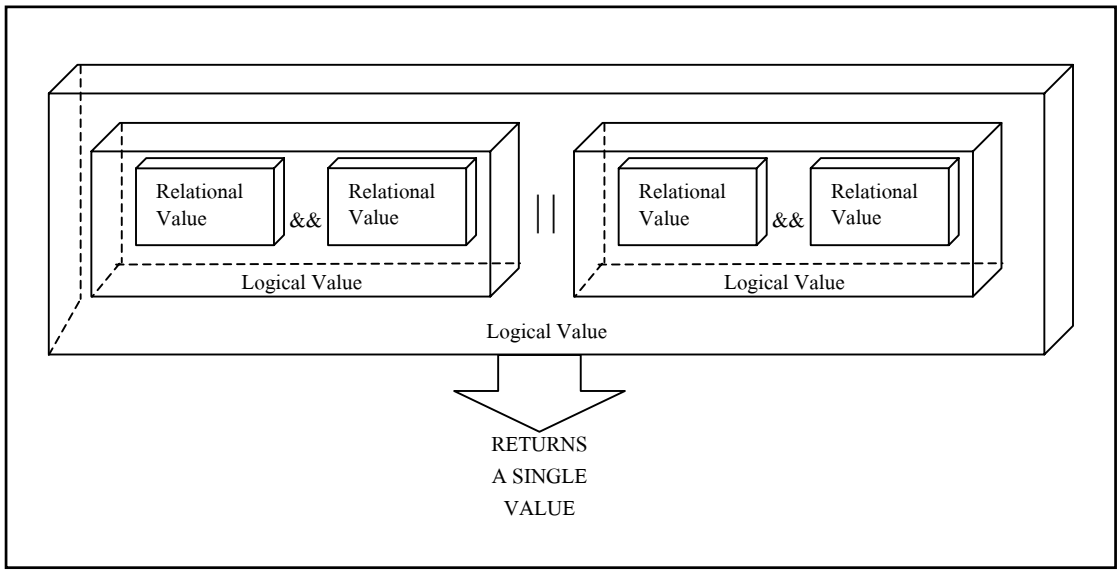


รูปที่ 5.10

ตัวอย่างโปรแกรมที่ 5.6 เป็นโปรแกรมที่มีตัวดำเนินการ AND และ OR มาใช้

```
#include <stdio.h>

main ()
{
    float number ;    /* User input number. */

    printf ( "\n\nGive me a number from 1 to 100 = > " ) ;
    scanf ( "%f", &number ) ;
                                                    <-- or
    if ( ( (number > = 1.0) && (number < = 10.0) ) | |
        ( (number > = 90.0) && (number < = 100.0) ) )
            printf ( "You gave me a number in the top or bottom 10%%." ) ;
}
```

ส่วนเงื่อนไขจะมีความซับซ้อนมากขึ้น ให้ย้อนกลับไปดูลำดับความสำคัญของเครื่องหมายตัวดำเนินการในบทก่อนหน้านี้ และเงื่อนไขสามารถแสดงได้ดังรูปที่ 5.11



รูปที่ 5.11

ตัวอย่างโปรแกรมที่ 5.7 เป็นโปรแกรมที่คำนวณปีที่เป็นปีอธิกสุทิน (leap year) ซึ่งเป็นปีที่มี 366 วัน

```
/* This program determines if a year is a leap year */
#include <stdio.h>

main ()
{
    int year, rem_4, rem_100, rem_400 ;
    printf ( “Enter the year to be tested :” ) ;
    scanf ( “%d”, &year ) ;

    rem_4 = year % 4 ;
    rem_100 = year % 100 ;
    rem_400 = year % 400 ;

    if ( (rem_4 = = 0  &&  rem_100  ! = 0)  ||  rem_400 = = 0 )
        printf ( “It’s a leap year. \n” ) ;
    else
        printf ( “Nope, it’s not a leap year. \n” ) ;
}
```

```
Enter the year to be tested: 1955
Nope, It’s not a leap year.
```

```
Enter the year to be tested: 2000
It’s a leap year.
```

```
Enter the year to be tested: 1800
Nope, It’s not a leap year.
```

ตัวอย่างโปรแกรมที่ 5.8 เป็นโปรแกรมที่ตรวจสอบว่าเป็นเลขคู่หรือเลขคี่

```
/* Program to determine if a number is even or odd */
#include <stdio.h>

main ()
{
    int number_to_test, remainder :

    printf ( “Enter your number to be tested :” ) ;
    scanf ( “%d”, &number_to_test ) ;

    remainder = number_to_test % 2 ;

    if ( remainder = = 0 )
        printf (“The number is even. \n” ) ;
    else
        printf (“The number is odd. \n” ) ;
}
```

```
Enter your number to be tested: 1234
The number is even.
```

```
Enter your number to be tested: 551
The number is odd.
```

5.2.1.3 รูปแบบทั่วไปของข้อความสั่ง if แบบมีมากกว่า 2 ทางเลือก แสดงได้ดังรูปที่ 5.12

```
if (condition1)
    statement1 ;
else if (condition2)
    statement2 ;
        !
else if (conditionn)
    statementn ;
else statement0 ;
```

รูปที่ 5.12

โดยถ้า condition1 มีค่าเป็นจริง (ค่าไม่ใช่ 0) แล้ว statement1 จะถูกกระทำการ และส่วนที่เหลือที่ใช้ในการตัดสินใจทั้งหมดจะถูกข้าม และถ้า condition1 มีค่าเป็นเท็จ (ค่าเป็น 0) แล้ว statement2 จะถูกกระทำการ และในทำนองเดียวกัน ถ้า condition2, condition3, …, conditionn เป็นเท็จแล้ว statement0 จะถูกกระทำการ

ในทำนองเดียวกัน statement1, statement2, …, statement0 อาจจะเป็น 1 ข้อความสั่ง หรือหลายข้อความสั่งก็ได้ ถ้ามีมากกว่า 1 ข้อความสั่ง ต้องถูกปิดล้อมด้วยเครื่องหมาย { และ }

ตัวอย่างโปรแกรมที่ 5.9 เป็นโปรแกรมที่ตรวจสอบจำนวนที่ป้อนเข้ามา ถ้ามีค่าน้อยกว่า 0 แสดงค่า –1, ถ้าเท่ากับ 0 ให้แสดงค่า 0 และนอกนั้นแสดงค่า 1

```
/* Program to implement the sign function */
#include <stdio.h>

main ()
{
    int number, sign ;

    printf ( "Please type in a number :" ) ;
    scanf ( "%d", &number ) ;

    if (number < 0)
        sign = –1 ;
    else if (number = = 0)
        sign = 0 ;
    else                    /* Must be positive */
        sign = 1 ;

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

```
Please type in a number: 1121
Sign = 1
```

```
Please type in a number: –158
Sign = –1
```

```
Please type in a number: 0
Sign = 0
```

ตัวอย่างโปรแกรมที่ 5.10 เป็นโปรแกรมที่แยกชนิดของตัวอักขระ เมื่อผู้ใช้ป้อนข้อมูลเข้ามา

```
/* This program categorizes a single character
        that is entered at the terminal                    */
#include <stdio.h>

main ()
{
    char c ;

    printf ( “Enter a single character : \n” ) ;
    scanf ( “%c”, &c ) ;

    if ( (c > = ‘a’  &&  c < = ‘z’)  | |  (c > = ‘A’  &&  c < = ‘Z’) )
        printf ( “It’s an alphabetic character. \n” ) ;
    else if ( c > = ‘0’  &&  c < = ‘9’ )
        printf ( “It’s a digit. \n” ) ;
    else
        printf ( “It’s a special character. \n” ) ;
}
```

```
Enter a single character:
&
It’s a special character.
```

```
Enter a single character:
8
It’s digit.
```

```
Enter a single character:
B
It’s an alphabetic character.
```

ตัวอย่างโปรแกรมที่ 5.11 เป็นโปรแกรมที่จะให้ผู้ใช้ป้อนข้อมูลเข้ามาในรูป ตัวถูกดำเนินการ, ตัวดำเนินการ และตัวถูกดำเนินการ

```
/* Program to evaluate simple expressions of the form
                number    operator    number                */
#include <stdio.h>

main ()
{
    float    value1, value2 ;
    char    operator ;

    printf ( “Type in your expression. \n” ) ;
    scanf ( “%f  %c  %f”,  &value1, &operator, &value2 ) ;

    if ( operator = = ‘+’ )
        printf ( “% . 2f\n”, value1 + value 2 ) ;
    else if ( operator = = ‘–’ )
        printf ( “% . 2f\n”, value1 – value 2 ) ;
    else if ( operator = = ‘*’ )
        printf ( “% . 2f\n”, value1 * value 2 ) ;
    else if ( operator = = ‘/’ )
        printf ( “% . 2f\n”, value1 / value 2 ) ;
}
```

```
Type in your expression.
123.5 + 59.3
182.80
```

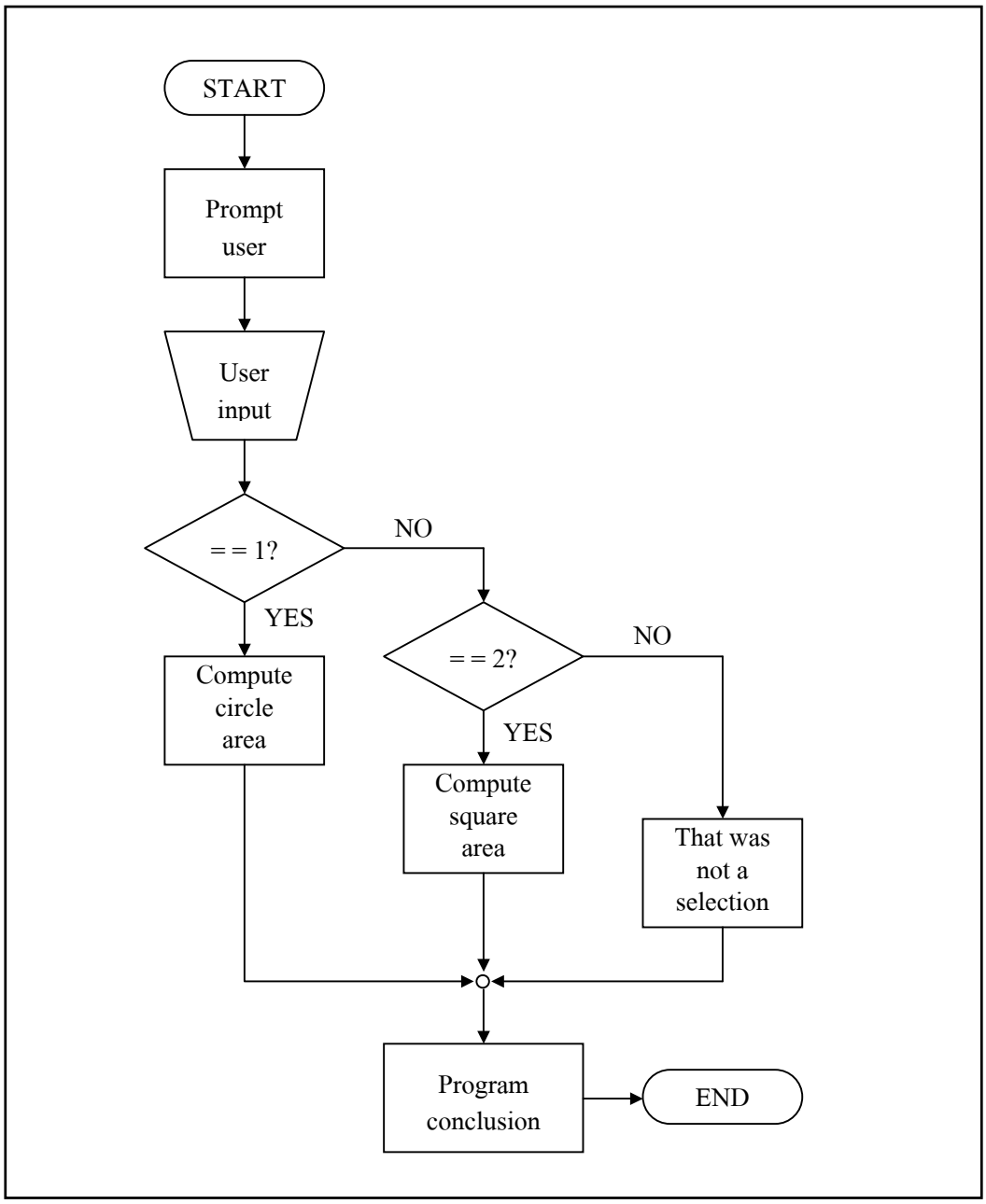
```
Type in your expression.
198.7 / 26
7.64
```

```
Type in your expression.
89.3 * 2.5
223.25
```

ตัวอย่างโปรแกรมที่ 5.12 เป็นโปรแกรมที่คำนวณหาพื้นที่วงกลม เมื่อผู้ใช้ป้อนตัวเลข 1 และคำนวณหาพื้นที่สี่เหลี่ยมจัตุรัส เมื่อป้อนตัวเลข 2

```
#include <stdio.h>
#define PI 3.141592

main ()
{
    float selection ;     /* User selection.              */
    float length ;        /* Length of side or radius.   */
    float area ;          /* Area in square units.       */

    printf ( "\n\nThis program will compute the area of\n" ) ;
    printf ( "a square or the area of a circle. \n" ) ;

    printf ( "\nSelect by number : \n" ) ;
    printf ( "1] Area of circle. 2] Area of square. \n" ) ;
    printf ( "Your selection (1 or 2) = >" ) ;
    scanf ( "%f", &selection ) ;

    if (selection = = 1)
    {
        printf ( "Give me the length of the circle radius = >" ) ;
        scanf ( "%f", &length ) ;

        area = PI * length * length ;
        printf ( "A circle of radius %f has an area of", length ) ;
        printf ( "%f square units.", area ) ;
    }
    else
    if (selection = = 2)
    {
        printf ( "Give me the length of one side of the square = >" ) ;
        scanf ( "%f", &length ) ;

        area = length * length ;
        printf ( "A square of length %f has an area of", length) ;
        printf ( "%f square units.", area ) ;
    }
    else
    {
        printf ( "That was not one of the selections. \n" ) ;
        printf ( "You must run the program again and\n" ) ;
        printf ( "select either a 1 or a 2. \n" ) ;
    }
    printf ( "\n\nThis concludes the program to calculate\n" ) ;
    printf ( "the area of a circle or a square." ) ;
}
```

และโปรแกรมนี้สามารถแสดงผังงานได้ดังรูปที่ 5.13

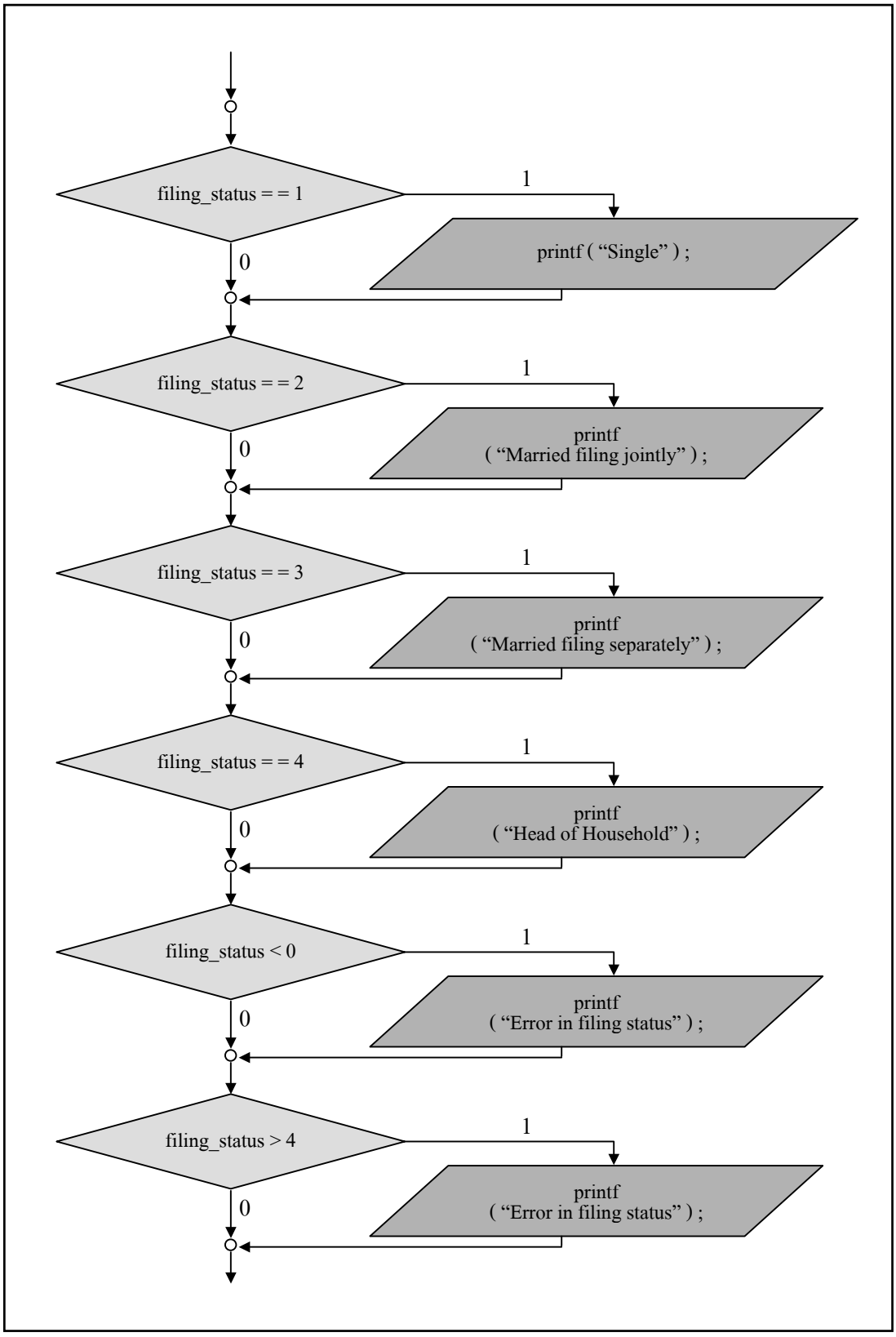


รูปที่ 5.13

ตัวอย่างโปรแกรมที่ 5.13 เป็นส่วนโปรแกรมที่มีหลายทางเลือก แต่ไม่ใช้ else

```
if (filing_status = = 1)
      printf ( “Single” ) ;
/* end if */

if (filing_status = = 2)
      printf ( “Married filing jointly” ) ;
/* end if */

if (filing_status = = 3)
      printf ( “Married filing separately” ) ;
/* end if */

if (filing_status = = 4)
      printf ( “Head of household” ) ;
/* end if */

if (filing_status < 0)
      printf ( “Error in filing status” ) ;
/* end if */

if (filing_status > 4)
      printf ( “Error in filing status” ) ;
/* end if */
```

จากส่วนโปรแกรมนี้สามารถแสดงผังงาน ได้ดังรูปที่ 5.14


filing_status = = 1
1
printf (“Single”) ;
0
filing_status = = 2
1
printf
(“Married filing jointly”) ;
0
filing_status = = 3
1
printf
(“Married filing separately”) ;
0
filing_status = = 4
1
printf
(“Head of Household”) ;
0
filing_status < 0
1
printf
(“Error in filing status”) ;
0
filing_status > 4
1
printf
(“Error in filing status”) ;
0

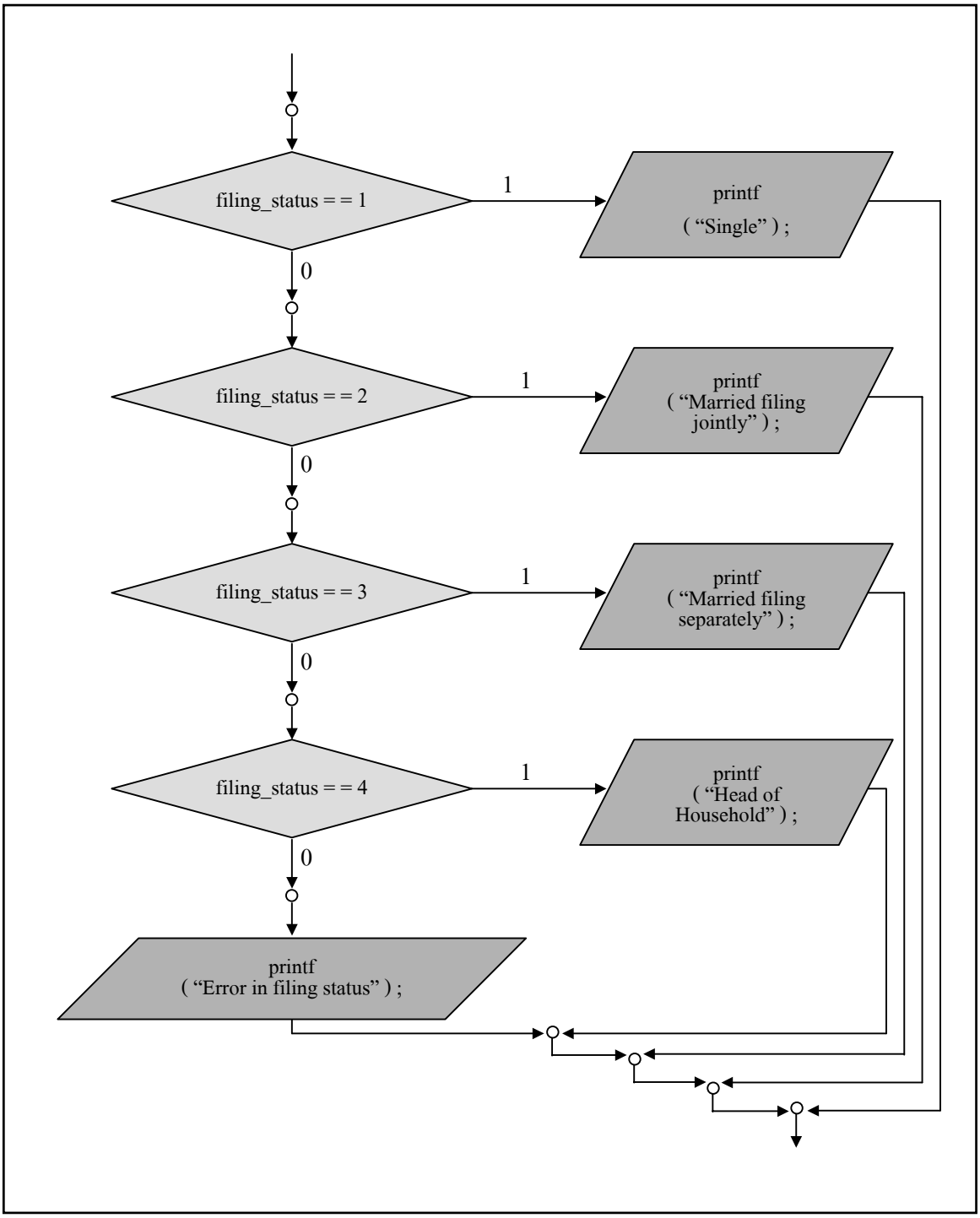
รูปที่ 5.14

จากการเขียนโปรแกรมและพิจารณาจากแผนผัง จะพบว่า ไม่มีการใช้ else จะพบว่า เมื่อเงื่อนไขด้านบนเป็นจริง ก็จะพิมพ์ข้อความ Single หลังจากนั้นก็จะมาตรวจสอบเงื่อนไขต่อมาอีก และจะทำไปเรื่อย ๆ จนถึงเงื่อนไขสุดท้าย ซึ่งเราจะพบว่า การเขียนโปรแกรมแบบนี้ เป็นการเขียนโปรแกรมที่ไม่มีประสิทธิภาพ เพราะว่าต้องเสียเวลาในการตรวจสอบเงื่อนไขด้านล่างอีก ในขณะที่เงื่อนไขด้านบนเป็นจริงแล้ว

ตัวอย่างโปรแกรมที่ 5.14 จะเหมือนกับโปรแกรมตัวอย่างที่ 5.13 เพียงแต่มีการนำ else มาใช้ ซึ่งจะทำให้โปรแกรมนี้มีประสิทธิภาพดีขึ้น กล่าวคือ เมื่อเงื่อนไขแรกเป็นจริง ก็จะพิมพ์ข้อความ Single หลังจากนั้นก็จะกระโดดมาที่ข้อความสั่งสุดท้ายถัดไป

```
if (filing_status = = 1)
        printf ( “Single” ) ;
else if (filing_status = = 2)
        printf ( “Married filing jointly” ) ;
else if (filing_status = = 3)
        printf ( “Married filing separately” ) ;
else if (filing_status = = 4)
        printf ( “Head of household” ) ;
else
        printf ( “Error in filing status” ) ;
/* end if */
```

และโปรแกรมนี้นำมาเขียนผังงาน ได้ดังรูปที่ 5.15


filing_status = = 1
1
printf
(“Single”) ;
0
filing_status = = 2
1
printf
(“Married filing
jointly”) ;
0
filing_status = = 3
1
printf
(“Married filing
separately”) ;
0
filing_status = = 4
1
printf
(“Head of
Household”) ;
0
printf
(“Error in filing status”) ;

รูปที่ 5.15

ตัวอย่างโปรแกรมที่ 5.15 จะเป็นโปแรกมที่มีลักษณะเหมือนตัวอย่างโปรแกรมที่ 5.13 คือไม่มี else

```
/* Program to determine if a number is even or odd */
#include <stdio.h>

main ()
{
    int number_to_test, remainder ;

    printf ( “Enter your number to be tested. :” ) ;
    scanf ( “%d”, &number_to_test ) ;

    remainder = number_to_tess % 2 ;

    if ( remainder = = 0 )
        printf ( “The number is even. \n” ) ;

    if ( remainder ! = 0 )
        printf ( “The number is odd. \n” ) ;
}
```

```
Enter your number to be tested: 2455
The number is odd.
```

```
Enter your number to be tested: 1210
The number is even.
```

5.2.1.4 รูปแบบทั่วไปของข้อความสั่ง if แบบซ้อน แสดงได้ดังรูป 5.16

แบบ 1

```
if (condition1)
    if (condition2)
        statement1 ;
    else
        statement2 ;
    else
        statement3 ;
```

ซึ่งจะมีความหมายเหมือนกับ

แบบ 2

```
if (condition1)  {
    if (condition2)
        statement1 ;
    else
        statement2 ;
}  else
        statement3 ;
```

รูปที่ 5.16

ข้อแนะนำ ควรใช้แบบ 2 เนื่องจากมีการใช้เครื่องหมาย { และ } ทำให้เข้าใจมากกว่า

ตัวอย่างโปรแกรมที่ 5.16 เป็นโปรแกรมที่มีการใช้เครื่องหมาย { และ }

```
if (number > 20) {
    if (number > 30)
        printf ( “Number between 21 and 29 inclusive\n” ) ;
    else
        printf (“Number exceeds 29\n” ) ;
}   else
        printf ( “Number does not exceed 20\n” ) ;
```

ในทำนองเดียวกัน

แบบ 3

```
if (condition1)
    statement1 ;
else
    if (condition2)
        statement2 ;
    else
        statement3 ;
```

ซึ่งจะมีความหมายเหมือนกับ

แบบ 4

```
if (condition1)
    statement1 ;
else {
    if (condition2)
        statement2 ;
    else
        statement3 ;  }
```

รูปที่ 5.17

ข้อแนะนำ ควรใช้แบบ 4 เนื่องจากมีการใช้เครื่องหมาย { และ }

ตัวอย่างโปรแกรมที่ 5.17 เป็นโปรแกรมที่มีการใช้เครื่องหมาย { และ }

```
if (ch > = ‘a’ && ch < = ‘z’)
    printf ( “Lowercase letter\n” ) ;
else  {
    if (ch > = ‘0’ && ch < = ‘9’)
        printf ( “Digit symbol\n” ) ;
    else
        printf ( “Other symbol\n” ) ;
}
```

ตัวอย่างโปรแกรมที่ 5.18 ถ้าเราเขียนส่วนของโปรแกรมดังนี้

```
if (condition1)
    if (condition2)
        statement1 ;
    else
        statement2 ;
```

จะทำให้เกิดความกำกวม กล่าวคือ else ควรจะจับคู่กับ if (condition1) หรือ if (condition2) ซึ่งสามารถเขียนได้ 2 แบบ ดังนี้

แบบ 1

```
if (condition1) {
    if (condition2)
        statement1 ;
    else
        statement2 ;
```

แบบ 2

```
if (condition1) {
    if (condition2)
        statement1 ;
    } else
        statement2 ;
```

กรณีเช่นนี้ จะมีหลักในการพิจารณาคือ else จะจับคู่กับ if ตัวที่อยู่ใกล้กับ else มากที่สุด จะยกตัวอย่าง 5.18 จะพบว่า จะมีความหมายเช่นเดียวกันกับ แบบ 1

ตัวอย่างโปรแกรมที่ 5.19 เป็นโปรแกรมที่มีข้อความสั่ง if ซ้อนอยู่

```
/* Program to evaluate simple expression of the form
                value     operator     value                    */
#include <stdio.h>

main ()
{
    float   value1, value2 ;
    char   operator ;

    printf ( "Type in your expression. \n" ) ;
    scanf ( "%f  %c  %f", &value1, &operator, &value2 ) ;

    if (operator = = '+')
        printf ( "% .2f\n", value1 + value2 ) ;
    else if (operator = = '–')
        printf ( " .2f\n", value1 – value2 ) ;
    else if (operator = = '*')
        printf ( "% .2f\n", value1 * value2 ) ;
    else if (operator = = '/')
        if (value 2 = = 0)
            printf ( "Division by zero. \n" ) ;
        else
            printf ( "% .2f\n", value1 / value2 ) ;
    else
        printf ( "Unknown operator. \n" ) ;
}
```

```
Type in your expression.
123.5 + 59.3
182.80
```

```
Type in your expression.
198.7 / 0
Division by zero.
```

```
Type in your expression.
125 $ 28
Unknown operator.
```

ตัวอย่างโปรแกรมที่ 5.20 เป็นส่วนของโปรแกรมทั้งสองที่มีความหมายเหมือนกัน

```
/* printf a message if all criteria are met. */
if (marital_status = = 'S')
    if (gender = = 'M')
        if (age > = 18  &&  age < = 26)
            printf ( "All criteria are met. \n" ) ;
```

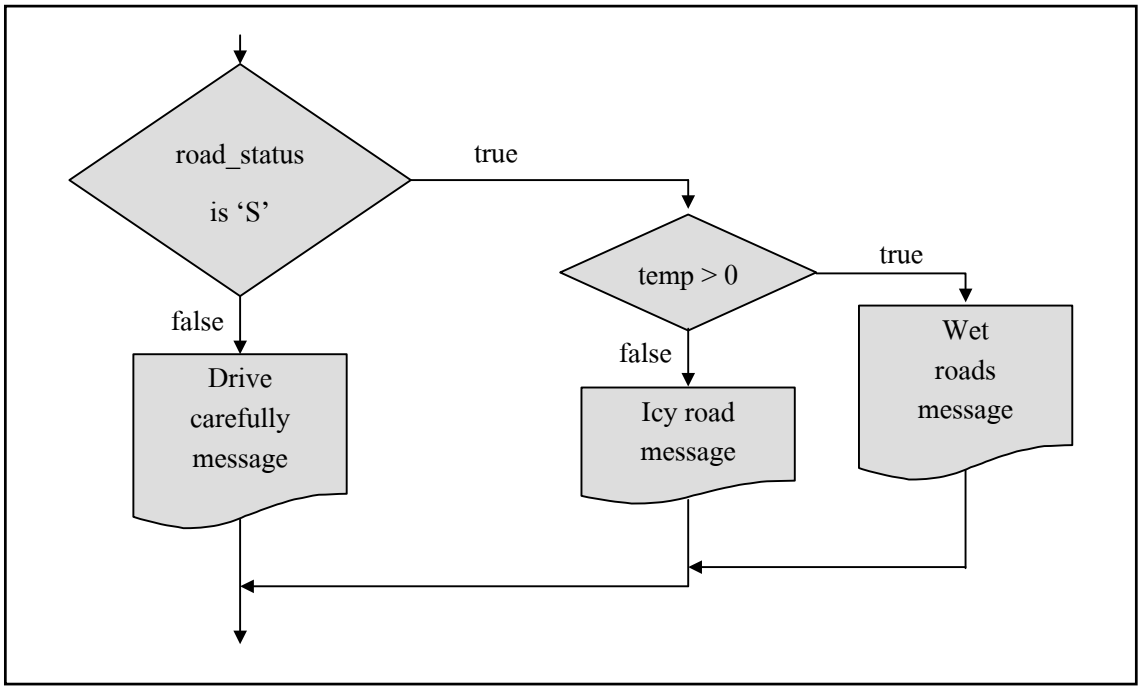
จะมีความหมายเหมือนกับ

```
if (marital_status = = 'S'  && gender = = 'M'
    &&  age > = 18  && age < = 26)
        printf ( "All criteria are met. \n" ) ;
```

ตัวอย่างโปรแกรมที่ 5.21 เป็นส่วนของโปรแกรมที่มีข้อความสั่ง if แบบซ้อน พร้อมทั้งแสดงผังงานด้วย

```
if (road_status = = 'S')
    if (temp > 0)  {
        printf ( "Wet roads ahead\n" ) ;
        printf ( "Stopping time doubled\n" ) ;
    }  else  {
        printf ( "Icy roads ahead\n" ) ;
        printf ( "Stopping time quardrupled\n" ) ;
    }
else
    printf ( "Drive carefully!\n" ) ;
```

จากส่วนของโปรแกรมนำมาแสดงผังงานได้ดังรูปที่ 5.17



รูปที่ 5.17

5.2.2 รูปแบบทั่วไปของตัวดำเนินการมีเงื่อนไข (?:)

```
Condition ? expression1 : expression 2
```

โดย condition จะเป็นนิพจน์ความสัมพันธ์

ถ้า condition มีค่าเป็นจริง ก็จะกระทำการ expression1

ถ้า condition มีค่าเป็นเท็จ ก็จะกระทำการ expression2

ในภาษา C ได้กำหนดตัวดำเนินการมีเงื่อนไข (?:) ให้มีความสัมพันธ์กับข้อความสั่ง if … else ดังนี้

เช่น x = (y < 0) ? – y : y ;

จะมีความหมายเหมือนกับ

```
if (y < 0)
   x = –y ;
else
   x = y ;
```

ตัวอย่างโปรแกรมที่ 5.22 เป็นโปรแกรมที่แสดงการใช้ตัวดำเนินการมีเงื่อนไข ถ้าผู้ใช้ป้อนตัวเลข 0 (เป็นเท็จ) ข้อความสั่ง printf ที่สอง ก็จะถูกกระทำการ

```
#include <stdio.h>

main ()
{
    int selection ;        /* User input selection */

    printf ( “Enter a 1 or a 0 = >” ) ;
    scanf ( “%d”, &selection) ;

    selection? printf (  “A one.” )  :  printf ( “A zero.” ) ‘
}
```

แต่ถ้าเราไปพบตัวดำเนินการเงื่อนไขที่มีความซับซ้อน เช่น

e1 ? e2 : e3 ? e4 : e5 จะหมายถึง e1 ? e2 : (e3 ? e4 : e5)

ตัวอย่างเช่น

```
sign = (number < 0) ? – 1 : ( (number = = 0) ? 0 : 1) ;
```

มีความหมายเหมือนโปรแกรม 5.9

5.2.3 รูปแบบทั่วไปของข้อความสั่ง switch

บางครั้งการใช้รูปแบบข้อความสั่ง if แบบหลายทางเลือก อาจจะซับซ้อนหรือยุ่งเหยิง ในภาษา C ได้กำหนดข้อความสั่งแบบ switch มาช่วยในการแก้ปัญหาดังกล่าว

รูปแบบทั่วไปของข้อความสั่ง switch แสดงได้ดังรูป 5.18

```
switch (expression)  {
    case constant–expression–1 :
        statement 1a ;
        statement 1b ;
        ------------
        ------------
    case constant–expression–2 :
        statement 2a ;
        statement 2b ;
        ------------
        ------------
------
------
case constant–expression–N :
    statement Na ;
    statement Nb ;
    ------------
    ------------
default :
    statement Da ;
    statement Db ;
    ------------
    ------------
}
```

รูปที่ 5.18

โดยข้อความสั่ง switch จะต้องประกอบด้วย

1) คำสงวน switch, case ส่วน default อาจจะมีหรือไม่มีก็ได้
2) expression จะถูกเรียกว่า นิพจน์ควบคุม (control expression) ซึ่งจะถูกปิดล้อมด้วยเครื่องหมาย (และ) โดยผลลัพธ์ที่ได้จะต้องมีค่าเป็นจำนวนเต็ม และมีข้อมูลเป็นชนิด int หรือ char เท่านั้น ห้ามเป็นชนิด double, float และสายอักขระ
3) ภายในเครื่องหมาย { และ } จะต้องมีคำสงวน case โดยแต่ละอนุประโยค (clause) ของ case ใด ๆ จะตามด้วยค่าคงตัว โดยค่าคงตัวนี้ อาจจะเป็นค่าจำนวนเต็มหรือค่าคงที่ของอักขระ ตามด้วยเครื่องหมาย colon (:) และตามด้วยข้อความสั่ง ซึ่งอาจจะเป็นข้อความสั่งว่าง (null statement) หรือ 1 ข้อความสั่ง หรือ มากกว่า 1 ข้อความสั่ง ถ้ามีมากกว่า 1 ข้อความสั่ง จะต้องถูกปิดล้อมด้วย { และ } และมีคำสงวน default (อาจจะมีหรือไม่มีก็ได้) ถ้ามี default จะต้องมีเครื่องหมาย colon (:) ตาม และข้อความสั่ง

ในข้อความสั่ง switch ส่วนของนิพจน์ควบคุม จะต้องถูกคำนวณก่อนเป็นอันดับแรก หลังจากนั้นก็จะนำค่าที่ได้ไปเปรียบเทียบดูว่าตรงกับค่าคงตัวของอนุประโยค ของ case ใด ถ้าตรงกัน ข้อความสั่งของอนุประโยค case นั้น จะถูกกระทำไปจนกระทั่งสุดข้อความสั่ง switch นั่นคือ พบเครื่องหมาย } ถ้าค่าของนิพจน์ควบคุมไม่ตรงกับค่าคงตัวของอนุประโยค ของ case ใดเลย ข้อความสั่งในส่วน default ก็จะถูกกระทำการ และถ้าไม่มี default การควบคุมก็จะออกจาก switch

ตัวอย่างโปรแกรมที่ 5.23 กำหนดค่า n = 2

```
n = 2 ;
switch (n)  {
    case   1 : printf ( “One\n” ) ;
    case   2 : printf ( “Two\n” ) ;
    case   3 : printf ( “Three\n” ) ;
    case   4 : printf ( “Four\n” ) ;
    default   : printf ( “Default\n” ) ;
}
printf ( “End of switch\n” ) ;
```

เมื่อ n = 2 แล้ว นิพจน์ควบคุม n จะตรงกับ case 2 ดังนั้น ข้อความสั่งในอนุประโยค case 2 จะถูกกระทำการไปตามลำดับ จนพบเครื่องหมาย } และได้ผลลัพธ์ ดังนี้

```
Two
Three
Four
Default
End of switch
```

ตัวอย่างโปรแกรมที่ 5.24 จะเหมือนกับตัวอย่าง 5.23 เพียงแต่ว่าอนุประโยค case 2 จะมีข้อความสั่งว่าง (null statement)

```
n = 2 ;
switch (n)  {
    case   1 : printf ( “One\n” ) ;
    case   2 :
    case   3 : printf ( “Two or Three\n” ) ;
    case   4 : printf ( “Four\n” ) ;
    default   : printf ( “Default\n” ) ;
}
printf ( “End of switch\n” ) ;
```

จะได้ผลลัพธ์ ดังนี้

```
Two or Three
Four
Default
End of switch
```

ตัวอย่างโปรแกรมที่ 5.25 เหมือนกับตัวอย่าง 5.23 เพียงแต่ว่าค่าของนิพจน์ควบคุมไม่ตรงกับอนุประโยคใด ๆ เลย ก็จะไปทำที่ default

เมื่อ n = 7

จะให้ผลลัพธ์ ดังนี้

```
Default
End of switch
```

แต่ถ้าไม่มี default ก็จะทำข้อความสั่งถัดจาก }

```
n = 7 ;
switch (n)  {
    case   1 : printf ( “One\n” ) ;
    case   2 :
    case   3 : printf ( “Two or Three\n” ) ;
    case   4 : printf ( “Four\n” ) ;
    default   : printf ( “Default\n” ) ;
}
printf ( “End of switch\n” ) ;
```

จะให้ผลลัพธ์ ดังนี้

```
End of switch
```

ตัวอย่างโปรแกรมที่ 5.26 เราจะพบว่า ถ้าค่าของนิพจน์ควบคุมตรงกับอนุประโยค case แรก ก็จะเริ่มทำข้อความสั่งแรกไปจนกระทั่งพบเครื่องหมาย } ดังโปรแกรมนี้

```
#include <stdio.h>

main ()
{
    char selection ;     /* User input selection. */

    printf ( "This program will show the formulas necessary\n" ) ;
    printf ( "to compute the power delivered by a voltage source\n" ) ;
    printf ( "to a series circuit consisting of three resistors\n" ) ;
    printf ( "when the value of the source voltage is know. \n" ) ;
    printf ( "\n\nSelect by letter : \n" ) ;
    printf ( "A] Resistor values known.      B] Total resistance know. \n" ) ;
    printf ( "C] Total current known. \n" ) ;
    printf ( "Your selection = >" ) ;
    scanf ( "%C", &selection ) ;

    switch (selection)
    {
        case 'A'   : printf ( "Rt = R1 + R2 + R3 \n" ) ;
        case 'B'   : printf ( "It =  Vt / Rt \n" ) ;
        case 'C'   : printf ( "Pt = Vt * Rt \n" ) ;
                     break ;
        default    : printf ( "That was not a correct selection." ) ;
    }   /* End of switch. */
}
```

ผลลัพธ์บนจอภาพจะมีลักษณะแตกต่างกัน ขึ้นอยู่กับผู้ใช้ป้อนข้อมูล คือ

1) ถ้าผู้ใช้ป้อนตัวอักษร A จะได้ผลลัพธ์ ดังนี้

```
Your selection = > A
Rt = R1 + R2 + R3
It  = Vt / Rt
Pt = Vt * It
```

2) ถ้าผู้ใช้ป้อนตัวอักษร B จะได้ผลลัพธ์ ดังนี้

```
Your selection = > B
It  = Vt / Rt
Pt = Vt * It
```

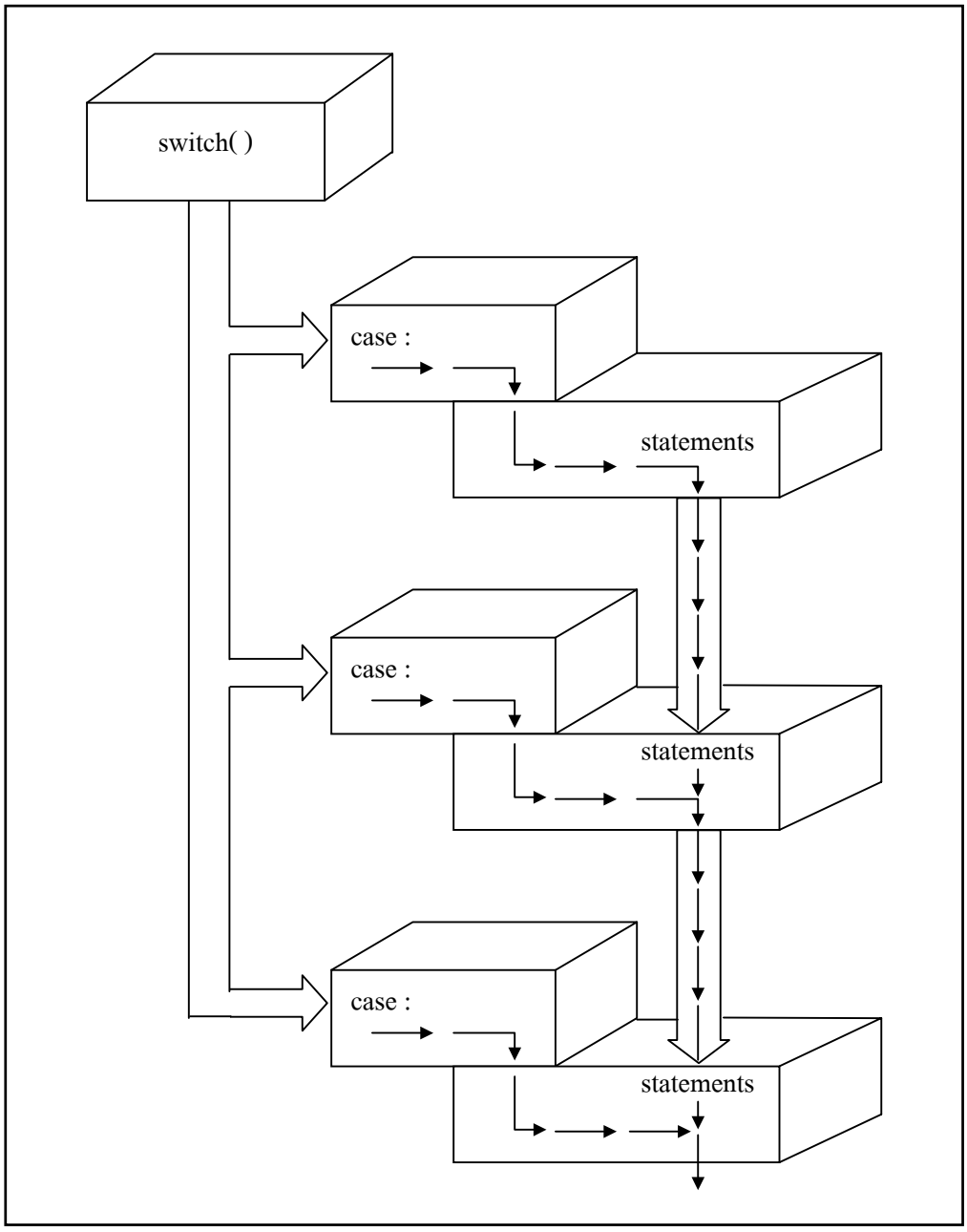
3) ถ้าผู้ใช้ป้อนตัวอักษร C จะได้ผลลัพธ์ ดังนี้

```
Your selection = > C
Pt = Vt * It
```

4) ถ้าผู้ใช้ป้อนตัวอักษร T จะได้ผลลัพธ์ ดังนี้

```
That was not a correct selection.
```

จากตัวอย่าง 5.23 ถึง 5.26 เราจะพบว่า ถ้าค่าของนิพจน์ควบคุมตรงกับอนุประโยค case ใด ก็จะกระทำการไปจนพบเครื่องหมาย } ซึ่งสามารถแสดงแผนภาพได้ดังรูปที่ 5.19



รูปที่ 5.19

5.2.4 ข้อความสั่ง break

จากตัวอย่างของการใช้ข้อความสั่ง switch ที่ผ่านมา จะพบว่า เมื่อนิพจน์ควบคุมมีค่าตรงกับอนุประโยค case ใด ๆ ใน switch แล้ว ก็จะเริ่มกระทำการข้อความสั่งหลัง case นั้นตามลำดับไปเรื่อย ๆ จนพบเครื่องหมาย } แต่ถ้าเราต้องการให้ทำเฉพาะข้อความสั่งหลัง case นั้นแล้วให้ออกจากข้อความสั่ง switch ให้เราเติมความสั่ง break หลังข้อความสั่งสุดท้ายของ case นั้น

ตัวอย่างโปรแกรมที่ 5.27

```
n = 3 ;
switch (n) {
    case   1 : printf ( “One\n” ) ;                 break ;
    case   2 :
    case   3 : printf ( “Two or Three\n” ) ;        break ;
    case   4 : printf ( “Four\n” ) ;                break ;
}
printf ( “End of switch\n” ) ;
```

จะให้ผลลัพธ์ ดังนี้

```
Two or Three
End of switch
```

จากตัวอย่างนี้ เมื่อนิพจน์ควบคุมมีค่าเท่ากับ 3 จะตรงกับ case 3 ก็จะทำข้อความสั่ง printf และทำข้อความสั่ง break เมื่อพบ break ก็จะหยุดและส่งการควบคุมออกนอกข้อความสั่ง switch และทำข้อความสั่งถัดจากเครื่องหมาย } (ให้ดูจากแผนภาพในตัวอย่าง จะมีลูกศรกำหนดทิศทาง)

จากตัวอย่างนี้ใน case 4 ไม่ต้องมี ข้อความสั่ง break ก็ได้ เนื่องจากเป็นข้อความสั่งสุดท้ายใน switch

เราจะพบว่า ข้อความสั่ง if แบบหลายทางเลือก จะมีความหมายเหมือนข้อความสั่ง switch ดังรูปที่ 5.20

```
switch (expression)
{
      case value1 :
          program statement
          program statement
              …
          break ;
      case value2 :
          program statement
          program statement
              …
          break ;
      case valuen :
          program statement
          program statement
              …
          break ;
      default :
          program statement
          program statement
              …
          break ;
}
```

เหมือนกับ

```
if (expression = = value1)
{
      program statement
      program statement
          …
}
else if (expression = = value2)
{
      program statement
      program statement
          …
}
  …
else if (expression = = valuen)
{
      program statement
      program statement
          …
}
else
{
      program statement
      program statement
          …
}
```

รูปที่ 5.20

ตัวอย่างโปรแกรมที่ 5.28 เป็นการแสดงความสัมพันธ์ของข้อความสั่ง if และ switch และไม่มีข้อความสั่ง break

```
switch (major_code)  {
      case 1 :
          printf (“ Student major is computer science.” ) ;
      case 7 :
          printf (“ Student major is computer engineering.” ) ;
      default :
          printf (“ Student major is a noncomputer field.” ) ;
}  /* end switch */
```

จะมีความหมายเหมือนกับ

```
if (major_code = = 1)  {
      printf (“Student major is computer science.” ) ;
      printf (“Student major is computer engineering.” ) ;
      printf (“Student major is a noncomputer field.” ) ;
}
else if (major_code = = 7)  {
      printf (“Student major is computer engineering.” ) ;
      printf (“Student major is a noncomputer field.” ) ;
}
else
      printf (“Student major is a noncomputer field.” ) ;
/* end if */
```

ถ้า major_code เท่ากับ 1 แล้ว ข้อความสั่งทั้ง 2 แบบนี้ จะให้ผลลัพธ์ดังนี้

Student major is computer science.

Student major is computer engineering.

Student major is a noncomputer field.

ตัวอย่างโปรแกรมที่ 5.29 จะเหมือนกับตัวอย่างโปรแกรมที่ 5.28 เพียงแต่มีข้อความสั่ง break

```
switch (major_code)  {
      case 1 :
          printf (“ Student major is computer science.” ) ;
          break ;
      case 7 :
          printf (“ Student major is computer engineering.” ) ;
          break ;
      default :
          printf (“ Student major is a noncomputer field.” ) ;
}  /* end switch */
```

จะมีความหมายเหมือนกับ

```
if (major_code = = 1)
      printf (“Student major is computer science.” ) ;
else if (major_code = = 7)
      printf (“Student major is computer engineering.” ) ;
else
      printf (“Student major is a noncomputer field.” ) ;
/* end if */
```

ตัวอย่างโปรแกรมที่ 5.30 เป็นโปรแกรมคำนวณการตัดเกรดโดยใช้ข้อความสั่ง switch เมื่อกำหนดข้อมูลดังตารางต่อไปนี้

semeter_average/10	Letter_grade
> = 9	A
8	B
7	C
6	D
< 6	E

และนำมาเขียนเป็นข้อความสั่ง switch ได้ดังนี้

```
switch (semester_average / 10)  {
      case 10 :
      case 9   :
          letter_grade = ‘A’ ;
          break ;
      case 8   :
          letter_grade = ‘B’ ;
          break ;
      case 7   :
          letter_grade = ‘C’ ;
          break ;
      case 6   :
          letter_grade = ‘D’ ;
          break ;
      default  :
          letter_grade = ‘F’ ;
}  /* end switch */
```

ตัวอย่างโปรแกรมที่ 5.31 เป็นการแสดงข้อความสั่ง switch ที่มีการรับข้อมูลจากแป้นพิมพ์ โดยให้ผู้ใช้ป้อนข้อมูลเข้ามาเก็บไว้ในตัวแปร character ที่มีชนิดข้อมูลแบบ char

```
switch (character)  {
      case ‘0’ : case ‘1’ : case ‘2’ : case ‘3’ : case ‘4’ :
      case ‘5’ : case ‘6’ : case ‘7’ : case ‘8’ : case ‘9’ :
          printf ( “The content is a decimal integer.” ) ;
          break ;
      default  :
          printf ( “The content is a nondecimal character.” ) ;
}  /* end switch */
```

จะพบว่ามีอนุประโยคทั้งหมด 8 case คือ case ‘0’ ถึง case ‘8’ จะมีข้อความสั่งว่าง เฉพาะ case ‘9’ จะมีข้อความสั่ง printf และ break จากข้อความสั่ง switch ดังกล่าว นำมาเขียนเป็นข้อความสั่ง if แบบหลายทางเลือกได้ดังนี้

```
if (character = = ‘0’)
      printf ( “The content is a decimal integer.” ) ;
else if (character = = ‘1’)
      printf ( “The content is a decimal integer.” ) ;
else if (character = = ‘2’)
      printf ( “The content is a decimal integer.” ) ;
else if (character = = ‘3’)
      printf ( “The content is a decimal integer.” ) ;
else if (character = = ‘4’)
      printf ( “The content is a decimal integer.” ) ;
else if (character = = ‘5’)
      printf ( “The content is a decimal integer.” ) ;
else if (character = = ‘6’)
      printf ( “The content is a decimal integer.” ) ;
else if (character = = ‘7’)
      printf ( “The content is a decimal integer.” ) ;
else if (character = = ‘8’)
      printf ( “The content is a decimal integer.” ) ;
else if (character = = ‘9’)
      printf ( “The content is a decimal integer.” ) ;
else
      printf ( “The content is a nondecimal character.” ) ;
/* end if */
```

จากข้อความสั่ง if นี้ สามารถเขียนโดยการนำตัวดำเนินการสัมพันธ์มาใช้ได้ดังนี้ คือ

```
if (character = = '0' || character = = '1' || character = = '2' ||
    character = = '3' || character = = '4' || character = = '5' ||
    character = = '6' || character = = '7' || character = = '8' ||
    character = = '9')
      printf ( "The content is a decimal interger." ) ;
else
      printf ( "The content is a nondecimal character." ) ;
/* end if */
```

ตัวอย่างโปรแกรมที่ 5.32 จะเหมือนกับตัวอย่างโปรแกรมที่ 5.11 เพียงแต่ว่าโปรแกรมนี้ใช้ข้อความสั่ง switch แทนข้อความสั่ง if

```
/* Program to evaluate simple expressions of the form
                value    operator    value                    */
#include <stdio.h>

main ( )
{
    float   value1, value2 ;
    char    operator ;

    printf ( "Type in your expression. \n" ) ;
    scanf ( "%f  %c  %f,  &value1, &operator, &value2 ) :

    switch (operator)
    {
        case '+' :
            printf ( "%. 2f\n", value1 + value2) ;
            break ;
        case '–' :
            printf ( "%. 2f\n", value1 – value2) ;
            break ;
        case '*' :
            printf ( "%. 2f\n", value1 * value2) ;
            break ;
        case '/' :
            printf ( "%. 2f\n", value1 / value2) ;
            break ;
        default :
            printf ( "Unknow operator. \n" ) ;
            break ;
    }
}
```

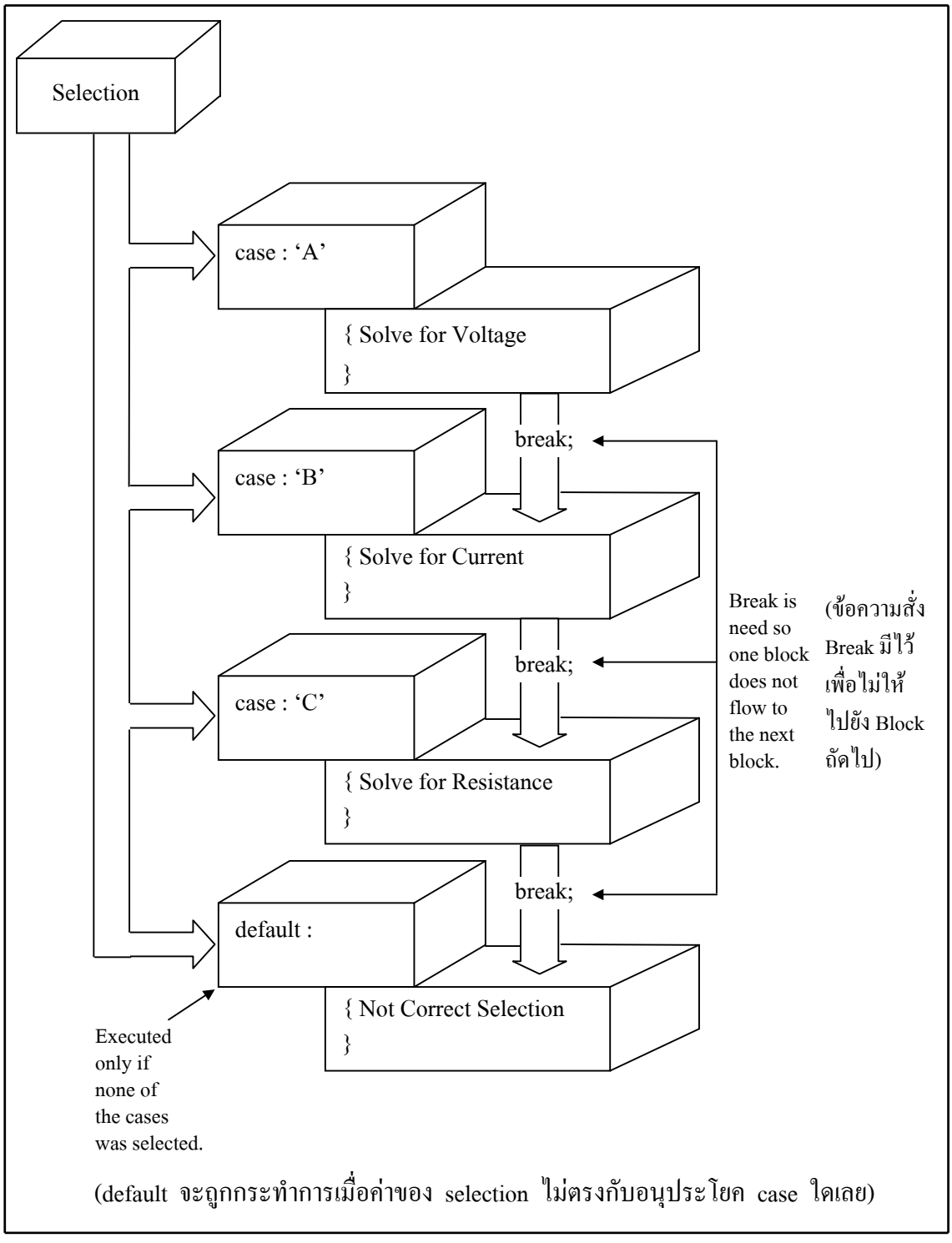
```
Type in your expression.
178.99 – 326.8
–147.81
```

ตัวอย่างโปรแกรมที่ 5.33 จะเป็นโปรแกรมที่แสดงการใช้เครื่องหมาย { และ } ปิดล้อมข้อความสั่งหลังอนุประโยค case ที่มีมากกว่า 1 ข้อความสั่ง

```
#include <stdio.h>

main ( )
{
    char   selection ;    /* Item to be selected by program user.   */
    float  voltage ;      /* Circuit voltage in volts.              */
    float  current ;      /* Circuit current in amps.               */
    float  resistance ;   /* Circuit resistance in ohms.            */

    printf ( "\n\nSelect the form of Ohm's Law needed by letter : \n" ) ;
    printf ( "A] Voltage  B] Current  C] Resistance \n" ) ;
    printf (" Your selection  (A, B, or C) = >" ) ;
    scanf ( "%c", &selection ) ;

    switch (selection)
    {
        case 'A'  :  {    /*Solve for voltage. */
                          printf ( "Input the current in amperes =>" ) ;
                          scanf ( "%f", &current ) ;
                          printf ( "Value of the resistance in ohms =>" ) ;
                          scanf ( "%f", &resistance ) ;
                          voltage = current * resistance ;
                          printf ( "The voltage is %f volts.", voltage ) ;
                     }
                     break ;
        case 'B'  :  {    /*Solve for current. */
                          printf ( "Input the voltage in volts =>" ) ;
                          scanf ( "%f", &voltage ) ;
                          printf ( "Value of the resistance in ohms =>" ) ;
                          scanf ( "%f", &resistance ) ;
                          current = voltage / resistance ;
                          printf ( "The current is %f amperes.", current ) ;
                     }
                     break ;
        case 'C'  :  {    /*Solve for resistance. */
                          printf ( "Input the voltage in volts = >" ) ;
                          scanf ( "%f", &voltage ) ;
                          printf ( "Value of the current in amperes = >" ) ;
                          scanf ( "%f", &current ) ;
                          resistance = voltage / current ;
                          printf ( "The resistance is %f ohms.", resistance ) ;
                     }
                     break ;
        default   :  printf ( "The was not a correct selection. \n" ) ;
                     prrintf ( "Please go back and select A, B, or C" ) ;
}   /* End of switch. */
```

ถ้าเงื่อนไข (selection = = ‘A’) เป็นจริงที่อนุประโยค case “A” ข้อความสั่งที่อยู่ภายในเครื่องหมาย { และ } จะถูกกระทำการ

จากโปรแกรมนี้ สามารถเขียนเป็นโครงสร้างทั่วไปดังรูปที่ 5.21



รูปที่ 5.21