บทที่ 4
โครงสร้างทางเลือก :IF และ SWITCH

วัตถุประสงค์

1. เพื่อให้นักศึกษาทราบถึงเงื่อนไขทางตรรก
2. เพื่อให้นักศึกษาสามารถกำหนดเงื่อนไขและนิพจน์ทางตรรกได้
3. เพื่อให้นักศึกษาทราบถึงนิพจน์บูลีน(bool)
4. เพื่อให้นักศึกษาสามารถแก้ไขปัญหาโปรแกรมที่มีเงื่อนไขและปฏิบัติงานเมื่อเงื่อนไขเป็นจริงได้ (if statement with a dependent statement)
5. เพื่อให้นักศึกษาสามารถแก้ไขปัญหาโปรแกรมที่มีเงื่อนไขและการปฏิบัติงานที่มีทางเลือกหลายทางได้ (if statement with two alternatives)
6. เพื่อให้นักศึกษาสามารถแก้ไขปัญหาโปรแกรมที่มีเงื่อนไขและทางเลือกหลายทางได้ (Multiple-Alternative Decision)
7. เพื่อให้นักศึกษาสามารถแก้ปัญหากับโปรแกรมที่มีการใช้คำสั่ง if และคำสั่ง switch ได้

สำหรับตำราเล่มนี้อธิบายถึงการออกแบบจากบนลงล่าง (top down design) ซึ่งเป็นเทคนิคที่มีการแบ่งปัญหาออกเป็นปัญหาย่อยๆให้แยกออกจากกันแบบบนลงล่าง โดยโปรแกรมจะประกอบด้วยโปรแกรมย่อยๆต่างๆที่มีการปฏิสัมพันธ์ต่อกัน โปรแกรมหนึ่งๆนั้นมีโครงสร้างการปฏิบัติการอยู่ 3 รูปแบบได้แก่ sequence , selection และ repetition สำหรับในบทที่ผ่านมาเป็นการแก้ปัญหาในลักษณะที่เป็นลำดับ(sequence) สำหรับในบทนี้จะกล่าวถึงโครงสร้างของโปรแกรมที่เป็นทางเลือก(selection) ในการทำงาน ซึ่งเกี่ยวข้องกับเงื่อนไขและนิพจน์ตรรกในการปฏิบัติงาน สำหรับในบทต่อไปจะกล่าวถึงโครงสร้างโปรแกรมในลักษณะของการกระทำซ้ำ

4.1 นิพจน์ตรรก (Logical Expressions)

ภาษา C++ นั้นใช้นิพจน์ทางตรรกเพื่อเลือกหนทางในการปฏิบัติงาน ดังเช่นตัวอย่างของคำสั่ง if ต่อไปนี้

```
if (weight > 100.00)
        ShipCost = 10.00 ;
Else
        ShipCost = 5.00 ;
```

โดยโปรแกรมจะเลือกคำสั่งใดทำงานนั้น จากการตรวจสอบถึงเงื่อนไข weight > 100.00 ถ้าเงื่อนไขจากการเปรียบเทียบค่ามีค่าเป็นจริง (true) จะเลือก ShipCost = 10.00; แต่ถ้าผลจากการเปรียบเทียบค่ามีค่าเป็นเท็จ (false) จะเลือก ShipCost = 5.00 ;

โดยทั่วไปการเปรียบเทียบค่านั้นสามารถใช้รูปแบบดังนี้

variable	relational-operator	variable
variable	relational-operator	constant
variable	equality-operator	variable
variable	equality-operator	constant

ตารางที่ 4.1 แสดงถึง relational และ equality operator และตารางที่ 4.2 แสดงถึงเงื่อนไขในการทำงานต่างๆ ถ้ากำหนดให้ x = -5 , power = 1024 , maxPower = 1024 , y = 7 , item = 1.5 , miniItem = -999.0 , momOrDad = 'm' , num = 999 และ sentinel = 999

ตารางที่ 4.1 relational and Equality Operators

Operator	Meaning	Type
<	less than	relational
>	greater than	relational
<=	less than or equal to	relational
>=	greater than or equal to	relational
==	equal to	Equality
!=	not equal to	Equality

ตารางที่ 4.2 ตัวอย่างเงื่อนไข

Operator	Condition	Value
<=	x <= 0	true
<	power < maxPower	false
>=	x >= y	false
>	item > miniItem	true
==	momOrDad == 'm'	true
!=	num != sentinel	false

ในกรณีที่เงื่อนไขทางตรรกมีการเปรียบเทียบมากกว่า 1 นิพจน์นั้น มี logical operator ที่ใช้สำหรับตรวจสอบค่า 3 ตัวกระทำได้แก่ && (and) , || (or) , ! (not) ดังเช่นตัวอย่างต่อไปนี้

```
(salary < minSalary) || (dependents > 5)
```

โดยตัวกระทำทางตรรก || (or) จะให้ผลลัพธ์จากการเปรียบเทียบเท่ากับ true ถ้านิพจน์ใดนิพจน์หนึ่งหรือทั้งสองนิพจน์มีค่าเท่ากับ true ดังตารางที่ 4.3

(temerature > 90.0) && (humidity > 0.90)

สำหรับตัวกระทำทางตรรก && จะให้ผลลัพธ์จากการเปรียบเทียบเท่ากับ true ถ้าทั้งสองนิพจน์มีค่าเท่ากับ true เท่านั้น ดังตารางที่ 4.4

ตารางที่ 4.3 ตัวกระทำ ||

Operand1	Operand2	Operand1 || Operand2
true	true	true
true	false	true
false	true	true
false	false	false

ตารางที่ 4.4 ตัวกระทำ &&

Operand1	Operand2	Operand1 && Operand2
true	true	true
true	false	false
false	true	false
false	false	false

สำหรับตัวกระทำ ! คือคอมพลีเมนต์ (complement) หรือไม่(negative) จะให้ผลลัพธ์ตรงข้าม ดังตารางที่ 4.5

ตารางที่ 4.5 ตัวกระทำ !

Operand	! Operand
true	false
false	true

ในการแก้ปัญหาโปรแกรมบ่อยครั้งต้องมีการตรวจสอบหรือคำนวณค่าหรือเปรียบเทียบค่าของข้อมูลและนิพจน์ต่างๆมากกว่า 2 นิพจน์ มีตัวกระทำต่างๆผสมผสานกัน ซึ่งตัวกระทำต่างๆในภาษา C++ นั้นมีลำดับความสำคัญในการทำงานที่ไม่เหมือนกัน ดังตารางที่ 4.6

ตารางที่ 4.6 Operator Precedence

Operator	Precedence	Description
! , + , -	Highest	Logical not , unary plus, unary minus
* , / , %		Multiplication, division , modulus
+ , -		Addition , subtraction
< , <= , >= , >		Relational inequality
== , !=		Equal ,not equal
&&		Logical and
||		Logical or
=	Lowest	Assignment

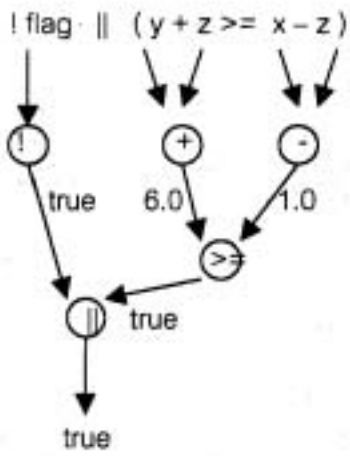
ตัวอย่างที่ 4.1 จงหาค่าของนิพจน์ทางตรรกต่อไปนี้

กำหนดให้

```
x    = 3.0 ;
y    = 4.0 ;
z    = 2.0 ;
flag = false ;
```

```
1. ! flag                      // ! false is true
2. x + y / z <= 3.5            // 5.0 <= 3.5 is false
3. ! flag || (y + z >= x - z)  // true || true is true
4. ! (flag || ( y + z >= x – z)  // ! (false || true) is true
```

รูปที่ 4.1 แสดงลำดับการทำงานทีละขั้นของการทำงาน ! flag || (y + z >= x - z)



flag	y	z	x
false	4.0	2.0	3.0

! flag || (y + z >= x – z)

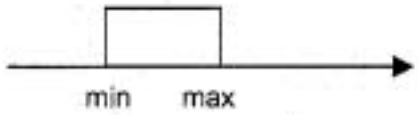
! false 4.0+2.0 3.0-2.0

true 6.0 >= 1.0

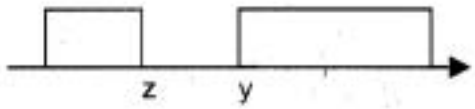
true || true

true

สำหรับการตรวจสอบเงื่อนไขที่เป็นช่วงของข้อมูลดังเช่นรูปข้างล่างนี้



โดยเราสามารถสร้างนิพจน์ตรรกเพื่อใช้สำหรับตรวจสอบค่าของตัวแปร x ว่ามีค่าอยู่ในช่วงระหว่าง min และ max โดยใช้ตัวกระทำ && ดังนี้ (min <= x) && (x <= max)
พิจารณารูปต่อไปนี้



ถ้าเราต้องการตรวจสอบค่าของตัวแปร X ว่าอยู่ในช่วงของข้อมูลดังรูปหรือไม่นั้นจะใช้ตัวกระทำ && ไม่ได้เพราะข้อมูลมิได้อยู่ในช่วงเดียวกัน แต่สามารถตรวจสอบโดยใช้ตัวกระทำ || แทน ดังนี้ (z > x) || ((x > y) ซึ่งค่าของนิพจน์จะมีค่าเท่ากับ true ถ้าข้อมูลเป็นจริงถ้าการเปรียบเทียบมีค่าเท่ากับ true เพียงค่าใดค่าหนึ่งเท่านั้น
ในการสร้างนิพจน์ทางตรรกต้องระมัดระวังถึงความผิดพลาดที่อาจเกิดขึ้นได้ ดังตัวอย่างที่ 4.2

ตัวอย่างที่ 4.2 แสดงนิพจน์ทางตรรกที่ผิด

```
x && y > z      // invalid logical expression
z <= x <= y
```

นอกจากการเปรียบเทียบค่าของข้อมูลที่เป็นตัวเลขแล้ว เรายังสามารถเปรียบเทียบค่าของข้อมูลที่เป็นตัวอักษรและข้อความได้ ซึ่งภาษา C++ นำค่า ASCII CODE ที่แทนตัวอักษรต่างๆนั้นมาใช้สำหรับเปรียบเทียบในกรณีที่เป็นข้อความจะเปรียบเทียบตัวอักษรทีละตัวจากซ้ายไปขวา ดังตารางที่ 4.8

ตารางที่ 4.8 ตัวอย่างการเปรียบเทียบตัวอักษรและข้อความ

Expression	Value
'a' < 'c'	true
'X' <= 'A'	false
'3' > '4'	false
('A' <= ch) && (ch <= 'Z')	true if ch contains an uppercase letter;otherwise false
"XYZ" <= "ABC"	false (X <= A)
"acts" > "aces"	true (t > e)
"ace" != "aces"	true(string are different)
"ace" < "aces"	true

จากตารางที่ 4.8 จะเห็นว่าการเปรียบเทียบนั้นจะเปรียบเทียบโดยนำตัวอักษรแต่ละตัวมาเปรียบเทียบกันถ้าเหมือนกันจะนำตัวอักษรลำดับถัดไปมาเปรียบเทียบ ในกรณีที่ข้อความมีจำนวนตัวอักษรไม่เท่ากันข้อความที่มีความยาวสั้นกว่าจะมีค่าน้อยกว่า ซึ่งเหมือนกับการเรียงคำต่างๆในพจนานุกรมเริ่มจากอักษร 'A' < 'B' < ... < 'Z' โดยตัวอักษรตัวเล็กมีค่ามากกว่าตัวอักษรตัวใหญ่ 'A' < 'a' ในกรณีที่เป็นตัวอักษรที่เป็นตัวเลข '0' < '1' < ... < '9'

นิพจน์บูลีน

ในการแก้ปัญหาโปรแกรมนั้นเราสามารถนำค่าของนิพจน์ทางตรรกให้แก่ตัวแปรชนิด bool ได้ตามรูปแบบดังนี้

```
variable = expression ;
```

เช่น
```
same = true ;
same = (x == y) ;
```
โดยค่า same จะมีค่าเท่ากับ true ถ้า x และ y มีค่าเท่ากัน

```
inRange = (n > -10 ) && ( n < 10) ;
```
ค่าของ inRange จะมีค่าเท่ากับ true ถ้าค่าของตัวแปร n มีค่าอยู่ระหว่าง –10 และ 10

```
isLetter = ( ('A' <= ch) && (ch <= 'Z')) || (('a' <= ch ) && (ch <= 'z')) ;
```
ค่าของ isLetter จะมีค่าเท่ากับ true ถ้าค่าของตัวแปร ch เป็นตัวอักษรภาษาอังกฤษตัวใหญ่หรืออักษรตัวเล็กก็ได้

```
even = ( n % 2 == 0) ;
```
ค่าของ even จะมีค่าเท่ากับ true ถ้า n เป็นเลขจำนวนเต็มคู่ เนื่องจาก เลขจำนวนเต็มคู่ทุกตัวเมื่อหารด้วย 2 แล้วจะเหลือเศษจากการหารเท่ากับ 0 เสมอ

สำหรับการแสดงผลของตัวแปรชนิด bool นั้นจะแสดงผลออกมาเป็นตัวเลข โดยถ้าค่าของตัวแปรชนิด bool เป็น true จะแสดงผลออกมาทางจอภาพมีค่าเท่ากับ 1 แต่ถ้ามีค่าเป็น false จะแสดงผลออกมามีค่าเท่ากับ 0

```
flag = false ;
cout << " The value of flag is " << flag ;
```

ผลของการทำงานจะแสดงข้อความดังนี้

```
The value of flag is 0
```

4.2 คำสั่ง if

การแก้ปัญหาโปรแกรมที่มีทางเลือกในการทำงานนั้น ภาษาC++ได้เตรียมคำสั่งให้กับโปรแกรมเมอร์ ไว้สำหรับในหลายๆรูปแบบคือ

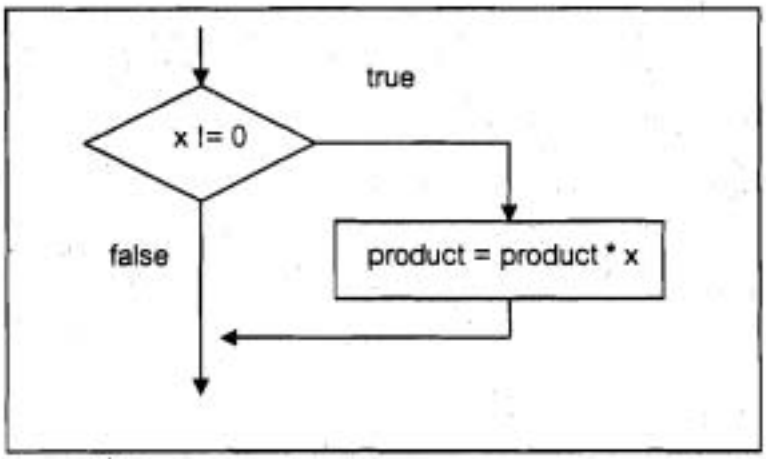
- if statement with a dependent statement
- if statement with two alternatives
- Multiple-Alternative Decision

If Statement with Dependent Statement

เป็นคำสั่งที่มีทางเลือกในการทำงาน 2 ทางแต่จะกระทำตามคำสั่งเมื่อเงื่อนไขเป็น true เท่านั้น แต่ถ้าเป็น false จะไม่มีการทำงานใดๆ

If Statement with Dependent Statement
รูปแบบ: if (condition) statement $_T$;
ตัวอย่าง: if (x != 0) product = product * x ;
ความหมาย : จะกระทำ statement $_T$ เมื่อ condition มีค่าเท่ากับ true ในกรณีอื่นๆจะข้ามไปทำคำสั่งต่อไป

รูปที่ 4.2 แสดงผังงาน (flowchart) ที่แสดงถึงขั้นตอนการประมวลผลของคำสั่ง if ทีละขั้นตอน โดยสี่เหลี่ยมข้าวหลามตัด(diamond-shaped box) แทนการตัดสินใจ(decision) ผลของการตัดสินใจจะให้ผลลัพธ์ได้ 2 ทางคือ true หรือ false เท่านั้น



รูปที่ 4.2 if statement with a dependent statement

ในกรณีที่ทางเลือกของการปฏิบัติงานนั้นมีการทำงานหลายๆคำสั่ง(Compound statements) ต้องเรียนคำสั่งเหล่านั้นภายใต้เครื่องหมาย { } ดังนี้

```
if  (popToday > popYesterday)
{
  growth = popToday – popYesterday ;
  growthPct = 100.0 * growth / popYesterday ;
  cout << "The growth percentage is " , growthPct ;
}
```

if Statement with Two Alternatives

เป็นคำสั่งที่มีทางเลือกในการทำงาน 2 ทาง โดยถ้าตรวจสอบเงื่อนไขเป็น true จะทำคำสั่งทางหนึ่ง และถ้าเงื่อนไขเป็น false จะกระทำตามคำสั่งอีกทางหนึ่ง

```
if (gross > 100.00 )
    net = gross – tax ;
else
    net = gross ;
```

if Statement with Two Alternatives

รูปแบบ:

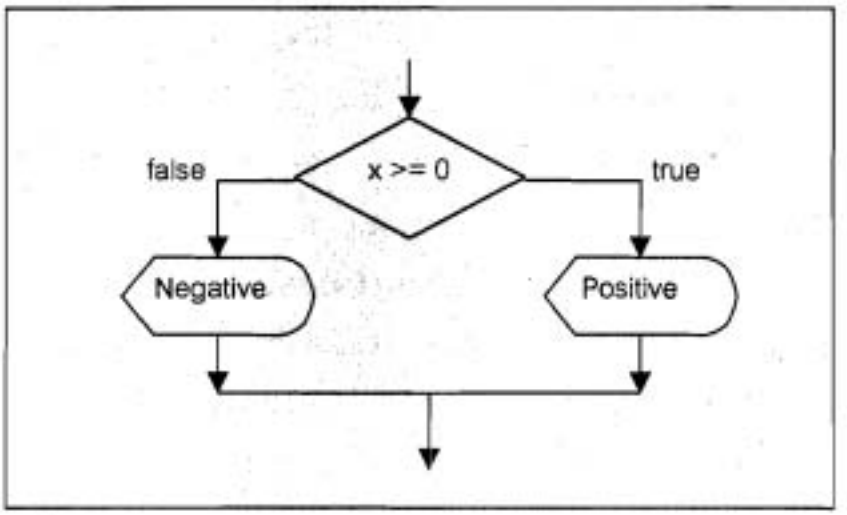
```
if (condition)
    statement T ;
else
    statement F ;
```

ตัวอย่าง:

```
if ( x >= 0.0 )
   cout << 'Positive" << endl ;
else
   cout <, "Negative" << endl ;
```

ความหมาย : จะกระทำ statement $_T$ เมื่อ condition มีค่าเท่ากับ true และกระทำ statement $_F$ เมื่อ condition เท่ากับ false

รูปที่ 4.3 แสดงผังงาน (flowchart) ของการตัดสินใจที่มีทางเลือกหลายทาง



กรณีที่แต่ละทางเลือกมีการปฏิบัติงานหลายๆคำสั่ง ต้องเขียนกลุ่มคำสั่งที่ปฏิบัติการภายใต้เครื่องหมาย { } ดังนี้

```
if (transactionType == 'c')
{
    cout << "Check for $" << transactionAmount << endl ;
    balance = balance – transactionAmount ;
}
else
{
    cout << "Deposit of $" << transactionAmount << endl ;
    balance = balance + transactionAmount ;
}
```

กรณีศึกษา :

โปรแกรมการคิดเงินเดือนของคนงาน

ปัญหา บริษัทแห่งหนึ่งต้องการคิดเงินเดือนให้กับคนงานในแต่ละสัปดาห์ โดยคนงานแต่ละคนมีอัตราค่าจ้างต่อชั่วโมง และจำนวนชั่วโมงทำงานแตกต่างกัน ถ้าคนงานคนใดที่ทำงานเกินสัปดาห์ละ 40 ชั่วโมงจำนวนชั่วโมงที่เกินบริษัทจะให้อัตราค่าจ้างเพิ่มจากเดิมเป็น 1.5 เท่าของอัตราปกติที่ได้รับ นอกจากนี้ถ้าคนงานคนใดที่มีรายได้ต่อสัปดาห์ตั้งแต่ 2500 บาทขึ้นไปจะต้องหักค่าภาษี ณ ที่จ่ายคนละ 50 บาท

วิเคราะห์ ต้องหาข้อมูลว่าอะไรคือข้อมูลเข้า อะไรคือผลลัพธ์ที่ต้องการ และต้องมีการประมวลผลอย่างไร ในที่นี้ข้อมูลเข้าคืออัตราค่าจ้างต่อชั่วโมง(rate) และจำนวนชั่วโมงที่คนงานทำงานต่อสัปดาห์(hours) ผลลัพธ์ที่ต้องการคือจำนวนเงินที่ได้รับในแต่ละสัปดาห์(gross) และจำนวนเงินที่ได้รับจริง(net)หลังจากหักภาษีแล้ว ซึ่งการทำงานนี้มีการกำหนดตัวแปรที่เก็บค่าคงที่หลายค่า ได้แก่ค่าภาษี(DUES) 50 บาท , จำนวนเงินได้ที่มากที่สุดที่ไม่ต้องหักภาษี

(MAX_NO_DUES) ,จำนวนชั่วโมงที่มากที่สุดที่ไม่ได้รับโบนัส(MAX_NO_OVERTIME) และอัตราของการทำงานล่วงเวลา(OVERTIME_RATE)

DATA REQUIREMENTS

Problem Constant

```
MAX_NO_DUES = 2500.00      // maximum earning without paying union dues
DUES = 50.00               // union dues to be paid
MAX_NO_OVERTIME = 40.0     // maximum hours without overtime pay
OVERTIME_RATE = 1.5        // time and a half for overtime
```

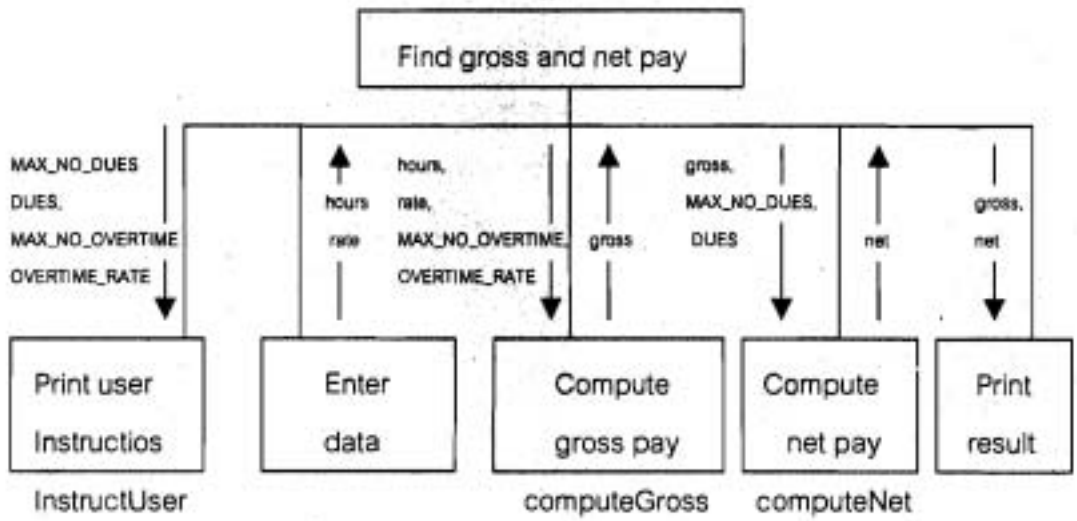
Problem Input

```
float  hours               // hours work
float rate                 // hourly rate
```

Problem Output

```
float gross                // gross pay
float net                  // net pay
```

ออกแบบ ขั้นตอนในการแก้ปัญหาแบ่งออกเป็น 5 ขั้นตอนใหญ่ ดังแสดงในรูปที่ 4.3



รูปที่ 4.3 ผังโครงสร้างของการแก้ปัญหาเงินเดือน

1. แสดงข้อความเพื่อแนะนำผู้ใช้งาน (ฟังก์ชัน instructUser)
2. ป้อนชั่วโมงทำงานและอัตราค่าจ้างต่อชั่วโมง
3. คำนวณหารายได้ (ฟังก์ชัน computeGross)
4. คำนวณหารายได้สุทธิ (ฟังก์ชัน computeNet)
5. แสดงรายได้ และรายได้สุทธิออกทางจอภาพ

เราสามารถสร้างฟังก์ชันขึ้นมาใหม่ ในที่นี้มี 3 ฟังก์ชันด้วยกันได้แก่ instructUser , computeGross และ computeNet โดยถูกเรียกใช้โดยฟังก์ชันหลัก

instructUser เป็นฟังก์ชันที่มีวัตถุประสงค์เพื่อแสดงสารสนเทศแก่ผู้ใช้งาน เพื่อให้ผู้ใช้ทราบถึงภาพรวมของโปรแกรมว่าทำอะไรและผู้ใช้ต้องป้อนข้อมูลอะไร ข้อมูลที่เกี่ยวข้องได้แก่ ค่าคงที่ต่างๆที่เรากำหนดขึ้น ซึ่งถือว่าเป็น global constants การออกแบบฟังก์ชันนี้จึงไม่มีอาร์กิวเมนต์ใดๆส่งให้แก่ฟังก์ชัน และไม่มีการส่งผลลัพธ์ใดๆกลับมา ดังนั้น

Input Arguments (none)

Function Return Value (none – a void function)

computeGross เป็นฟังก์ชันคำนวณรายได้ต่อสัปดาห์ของคนงาน ดังนั้นจำเป็นต้องทราบอัตราค่าจ้างต่อชั่วโมง(hours)และจำนวนชั่วโมงทำงานต่อสัปดาห์(rate) นอกจากนี้ต้องทราบค่าของจำนวนชั่วโมงที่มากที่สุดที่ไม่ได้รับโบนัส(MAX_NO_OVERTIME) และ อัตราของการทำงานล่วงเวลา(OVERTIME_RATE) เพื่อนำไปใช้ในการคำนวณ

Input Arguments

float hours // number of hours worked

float rate

Function Return Value

float gross

FORMULA

Gross pay = Hours worked * Hourly pay

ในการคำนวณนั้นเนื่องจากมีเงื่อนไขของรายได้ที่ได้รับซึ่งต้องมีการตรวจสอบถึงจำนวนชั่งโมงการทำงาน ในกรณีที่เกิน 40 ชั่วโมงต้องมีการปรับเปลี่ยนอัตราค่าจ้างใหม่ ดังนั้นจึงมีการกำหนดตัวแปรใหม่เพื่อใช้ในการทำงานภายในฟังก์ชันดังนี้

```
Local Data for computeGross
float gross          // gross pay
float regularPay     // pay for first 40 hours of work
float overtimePay    // pay for hours in excess of 40

Algorithm for computeGross
If the hour worked excedds 40.0 (max hours before overtime)
        Compute regularPay
        Compute overtimePay
        Add regularPay to overtimePay to et ross
else
        Compute ross as hour * rate
```

computeNet เป็นฟังก์ชันในการหารายได้สุทธิหลังจากหักภาษี ซึ่งจำเป็นต้องทราบรายได้ทั้งหมดเสียก่อน เราจะนำรายได้ทั้งหมดมาเปรียบเทียบกับรายได้ต่ำสุดที่ไม่ต้องเสียภาษี ซึ่งสามารถวิเคราะห์และออกแบบฟังก์ชันนี้ได้ดังนี้

```
Input Arguments
float  gross         // gross pay

Function return Value
float net            // net pay
```

FORMULA

net pay = gross pay – deductions

การทำงานในฟังก์ชันจำเป็นต้องมีการประกาศตัวแปรภายใต(local variable) เพื่อใช้งานดังนี้

Local Data for computeNet

float net

Algorithm for computeNet

if the gross pay is larger than 100.00

Deduct the dues of 50.00 from gross pay

else

Deduct no dues

ตัวอย่างที่ 4.3 โปรแกรมหาเงินเดือนคนงาน

```
// File: payrollFunctions.cpp
// Computes and displays gross pay and net pay given an hourly
//        rate and number of hours worked. Deducts union dues of 50.00
//        if gross salary excessed 2500.00; otherwise, deducts no dues

#include <iostream>
using namespace std;

// Function used
void instructUser () ;
float computeGross (float , float) ;
float computeNet (float) ;
```

```
const float  MAX_NO_DUES = 2500.00 ;        // max earning before dues
const float DUES = 50.00 ;                  // dues amount
const float MAX_NO_OVERTIME = 40.00 ;       // max hours before overtime
const float OVERTIME_RATE = 1.5 ;           // overtime rate

int main ( )
{
    float  hours ;          // input : hours worked
    float rate ;            // input : hourly pay rate
    float gross ;           // output : gross pay
    float net ;             // output : net pay

    // Display user instructions
    instructUser ( ) ;

    // Enter hours and rate
   cout << "Hours worked : " ;
   cin >> hours ;
   cout << " Hourly rate : " ;
   cin >> rate ;

   // Compute gross salary
   gross = computeGross (hours , rate);
   // Compute net salary
   net = computeNet (gross) ;
   // Print gross and net
   cout << "Gross salary is " << gross << endl ;
```

```
    cout << " Net salary is " <, net << endl ;

    return 0;
}

// Displays user instructions
void instructUser ( )
{
    cout << " This program computes gross and net salary. " << endl ;
    cout << " A dues amount of " << DUES << " is deducted for " <, endl;
    cout << "an emplayee who raens more than " << MAX_NO_DUES << endl <<endl;
    cout << " Overtime is paid at the rate of " << OVERTIME_RATE << endl;
    cout << "times the regular rate for hours worked over "
         << MAX_NO_OVERTIME << endl << endl ;
    cout << "Enter hours worked and hourly rate " << endl ;
    cout << "pn separate lines after the prompts. " << endl ;
    cout << "Press <return> after typing each number. " << endl << endl;
}  // end instructUser

// Find the gross pay
float computeGross ( float hours ,                  // IN : number of hours worked
                     float rate)                    // IN : hourly pay rate
{
    // Local data
    float gross ;               // RESULT : gross pay
    float regularPay ;          // pay for first 40 hours
    float overtimePay ;         // pay for hours in exvess of 40
```

```
    // Compute gross pay.
    if (hours > MAX_NO_OVERTIME)
    {
        regularPay = MAX_NO_OVERTIME * rate ;
        overtimePay = (hours – MAX_NO_OVERTIME) * OVERTIME_RATE * rate ;
        gross = regularPay + overtimePay ;
    }
    else
        gross = hours * rate ;

    return gross ;
} // end computeGross

// Find the net pay
float computeNet (float gross)                  // IN: gross salary
{
    // Local data
    float net ;                                 // RESULT : net pay

    // Compute net pay
    if (gross > MAX_NO_DUES)
        net = gross – DUES ;                    // deduct dues amount
    else
        net = gross ;                           // no deductions
    return net
}
// end computeNet
```

การทดสอบ โดยป้อนข้อมูลที่ทำให้โปรแกรมถูกทดสอบทุกๆทางเลือก ในที่นี้ทดลองป้อนจำนวนชั่วโมงทำงานเท่ากับ 50 และอัตราค่าจ้างต่อชั่วโมงเท่ากับ 150 บาท

```
This program computes gross and net salary.
A dues amount of 50.00 is deducted for
an employee who earns more than 2500.00
Overtime is paid at the rate of 1.5
times the regular rate on hours worked over 40
Enter hours worked and hourly rate
on separate lines after the prompts.
Press <return> after typing each number.

Hours worked : 50
Hourly rate : 150
Gross salary is 9000.00
Net salary is 8950.00
```

Multiple-Alternative Decisions

การแก้ไขปัญหาโปรแกรมที่มีการใช้คำสั่ง if ซ้อน if นั้นค่อนข้างยุ่งยาก เช่น

```
if (x>0)
    numPos = numPos +1 ;
else
    if ( x<0)
        numNeg = numNeg + 1 ;
    else
        numZero = numZero + 1 ;
```

จากตัวอย่างข้างต้นถ้าผู้เขียนโปรแกรมมีมาตรฐานในการเขียนคำสั่งไม่ดีไม่มีการเยื้องที่เหมาะสมการแก้ไขหรือติดตามการทำงานจะเป็นสิ่งที่ยุ่งยากมากไม่ทราบว่า else ตัวไหนเป็นของเงื่อนไขของ if ใด การแก้ไขถ้าเลือกที่จะแก้ปัญหาโดยใช้ คำสั่ง if ในการตรวจสอบเป็นลำดับโดยประกอบด้วยคำสั่ง if ถึง 3 คำสั่ง ดังนี้

```
// Less efficient sequence of if statement
        if ( x>0)
            numPos = numPos + 1;
        if ( x<0)
            numNeg = numNeg + 1;
        if ( x == 0)
            numZero = numZero + 1;
```

ถึงแม้การทำงานจะเข้าใจได้ง่ายแต่โปรแกรมไม่มีประสิทธิภาพดีเพียงพอเนื่องจากการตรวจสอบอาจเป็นเพียงกรณีแรกที่เกิดขึ้นบ่อยๆ แต่โปรแกรมต้องตรวจสอบอีก 2 กรณีทุกๆครั้ง ทำให้เสียเวลาในการทำงาน ภาษา C++ ได้เตรียมรูปแบบของการตัดสินใจหลายทางเลือกเพื่อแก้ปัญหาต่างๆเหล่านี้ดังรูปแบบดังนี้

Multiple-Alternative Decision Form
รูปแบบ: if (condition $_1$) statement $_1$; else if (condition $_2$) statement $_2$; ..... else if (condition $_n$) statement $_n$;

ดังนั้นการตรวจสอบค่าของตัวเลขว่าเป็นเลขจำนวนเต็มบวก หรือเลขจำนวนเต็มลบ หรือค่าศูนย์สามารถเขียนคำสั่งได้ดังนี้

```
if ( x>0 )
    numPos = numPos = 1 ;
else if ( x<0 )
    numNeg = numNeg + 1;
else
    numZero = numZero + 1;
```

การทำงานจะตรวจสอบเงื่อนไขแรกก่อน ถ้ามีค่าเป็น true จะทำงานตามคำสั่งที่กำหนด และข้ามเงื่อนไขที่เหลือไปทำคำสั่งถัดไป เราถือว่าคำสั่ง if เป็น1คำสั่ง แต่ถ้าตรวจสอบเงื่อนไขแล้วมีค่าเป็น false จะตรวจสอบเงื่อนไขในลำดับถัดมาซึ่งถ้าเป็น true จะกระทำตามคำสั่งที่กำหนดแต่ถ้าไม่ใช่จะตรวจสอบคำสั่งในลำดับถัดมา จนกระทั่งถึง else สุดท้ายจะเป็นคำสั่งที่ปฏิบัติงานถ้าเงื่อนไขเป็น false ทุกเงื่อนไข

ตัวอย่างที่ 4.4 ฟังก์ชันในการแสดงเกรด

สมมุติว่าต้องการพิมพ์เกรดออกทางจอภาพ โดยตรวจสอบคะแนนสอบที่ได้รับมา มีเงื่อนไขดังนี้

คะแนนสอบ	เกรดที่ได้รับ
90 และมากกว่า	A
80 ถึง 89	B
70 ถึง 79	C
60 ถึง 69	D
ต่ำกว่า 60	F

เห็นได้ว่าเป็นการทำงานที่มีทางเลือกหลายทางซึ่ง เราสามารถนำมาสร้างเป็นฟังก์ชันได้ดังนี้

```
// Displays  the letter grade corresponding to an exam
// score assuming a normal scale.
```

```
void displayGrade (int score)
{
    if (score >= 90 )
        cout << "Grade is A " << endl ;
    else if (score >= 80 )
        cout << "Grade is B " << endl ;
    else if (score >= 70 )
        cout << "Grade is C " << endl ;
    else if (score >= 60 )
        cout << "Grade is D " << endl ;
    else
        cout << "Grade is F " << endl ;
}
```

การเรียกใช้เพียงแต่ส่งผ่านคะแนนใดๆให้แก่ฟังก์ชันเช่น displayGrade(75) ; โดยค่า 75 เป็นค่าของคะแนนที่ส่งให้แก่ตัวแปร score ในฟังก์ชัน ผ่านกระบวนการตรวจสอบเป็นลำดับจนกระทั่งพบเงื่อนไขที่มีค่าเป็น true นั่นคือ score >= 70 ดังนั้นการทำงานจะพิมพ์ "Grade is C" ออกทางจอภาพ และจบการทำงานของฟังก์ชัน

ตัวอย่างที่ 4.5 ฟังก์ชันในการคำนวณภาษี

ในการคำนวณภาษีรายได้บุคคลนั้น อัตราการคิดภาษีขึ้นอยู่กับเงินเดือน สมมุติว่ามีอัตราเป็นดังตารางข้างล่างนี้

Range	Salary	Base Tax	Percentage of Express
1	0.00 ถึง 14,999.99	0.00	15
2	15,000.00 ถึง 29,999.99	2250.00	16
3	30,000.00 ถึง 49,999.99	4650.00	18
4	50,000.00 ถึง 79,999.99	8250.00	20
5	80,000.00 ถึง 150,000.00	14,250.00	25

ถ้าเงินเดือนนั้นอยู่ในช่วงใด จะนำภาษีที่อยู่ในช่วงที่ต่ำกว่ามาบวกกับอัตราที่ต้องเสียในช่วงนั้นๆ ซึ่งเราสามารถแก้ปัญหานี้ได้ดังนี้

```
// Compute the tax due based on a tax table
float computeTax (float salary)
{
    if (salary < 0.00 )
        tax = -1 ;
    else if (salary < 15000.00)                    // first range
        tax = 0.15 * salary ;
    else if (salary < 30000.00)                    // second range
        tax = (salary – 15000.00) * 0.16 + 2250.00 ;
    else if (salary < 50000.00)                    // third range
        tax = (salary – 30000.00 ) * 0.18 + 4650 .00 ;
    else if (salary < 80000.00)                    // fourth range
        tax = (salary – 50000.00) * 0.20 + 8250.00 ;
    else if (salary <=150000.00)                   // fifth range
        tax = (salary – 80000.00) * 0.25 + 14250.00 ;
    else
        tax = -1 ;

    return tax ;
}
```

การเรียกใช้งานนั้นเนื่องจากฟังก์ชันนี้มีการส่งผ่านค่ากลับ 1 ค่าดังนั้นต้องมีตัวแปรซึ่งมีชนิดเป็น float รับค่า ดังเช่น

```
Tax = computeTax(20000.00 ) ;
```

การทำงานค่าของ salary ในฟังก์ชันมีค่าเท่ากับ 20000.00 ต่อจากนั้นจะตรวจสอบเงื่อนไขและกระทำงานตามคำสั่งเมื่อเงื่อนไขเป็น true ในที่นี้จะส่งค่า 3050.00 กลับมาให้ค่าแก่ตัวแปร Tax

ตัวอย่างที่ 4.6 โปรแกรมหาค่าที่มากที่สุดของเลข 3 จำนวนใด ๆ

โปรแกรมต่อไปนี้เป็นการหาค่าจำนวนที่มากที่สุดของเลข 3 จำนวนใดๆ โดยมีการสร้างฟังก์ชันในการเปรียบเทียบข้อมูลใดๆ 3 จำนวน ชื่อ compareThree ภายในฟังก์ชันมีการเรียกใช้ฟังก์ชัน larger ในการเปรียบเทียบค่า 2 ค่า ดังนี้

```
// Program : Largest of three numbers
#include <iostream>
using namespace std;
double larger(double x , double y);
double comparethree(double x ,double y , double z);

int main()
{
 double one ,two ;

 cout << "The larger of 5 and 10 is " << larger(5 , 10 ) << endl;
 cout << "Enter two numbers :" ;
 cin >> one >> two ;
 cout << endl ;
 cout << "the larger of " << one <<"and " << two << "is: << larger(one ,two) << endl;
 cout << "The largers of 23 , 34 , and 12 is " << compareThree(23 , 34 , 12 ) << endl ;

 return 0 ;
}
```

```
double larger(double x , double y )
{
   if ( x>= y )
        return x ;
   else
        return y ;
}

double compareThree(double x , double y , double z)
{
   return larger( x, larger (y , z)) ;
}
```

ทดสอบ

```
The larger of 5 and 10 is 10
Enter two numbers : 25  73

The larger of 25 and 73 is 73
The largest of 23 , 34 , and 12 is 34
```

การทำงานของโปรแกรมนี้มีการเรียกฟังก์ชันภายในฟังก์ชัน โดยภายในฟังก์ชัน comparethree มีการเรียกใช้ฟังก์ชัน larger 2 ครั้งการทำงานจะกระทำฟังก์ชันที่เรียกในวงเล็บในสุดก่อน เมื่อมีการส่งผ่านค่ากลับมาจากการทำงาน จะมีการเรียกใช้ฟังก์ชัน larger ในครั้งที่ 2

ฟังก์ชัน larger เป็นฟังก์ชันในการเปรียบเทียบค่าที่มากที่สุดของเลข 2 จำนวนใดๆ ส่งกลับมายังจุดเรียกใช้ ดังนั้นเมื่อมีการเรียกใช้ compareThree(23, 34 , 12) การทำงานจะเรียกใช้ larger (34 , 12) ซึ่งอยู่ในวงเล็บในสุดก่อนผลของการทำงานจะโดยส่งผ่านค่า 34 มายังจุดเรียก

เรียกใช้ ต่อจากนั้นจะมีการเรียกใช้ฟังก์ชัน larger (23 , 34) เป็นครั้งที่ 2 ซึ่งการทำงานจะส่งผ่านค่า 34 กลับมา ซึ่งค่านี้จะถูกส่งกลับไปยังฟังก์ชันหลัก มีผลให้ผลลัพธ์ของการเรียกฟังก์ชัน compareThree(23 , 34 , 12) มีค่าเท่ากับ 34

ตัวอย่างที่ 4.7 การเรียงลำดับเลข 3 จำนวนใดๆ

สำหรับตัวอย่างนี้จะมีการสร้างฟังก์ชันชื่อ order เพื่อทำหน้าที่ในการนำเลข 2 จำนวนใดๆ มาตรวจสอบเพื่อเรียงลำดับ เป็นการแสดงถึงการพารามิเตอร์ที่ทำหน้าที่ทั้งเป็น input และ output

```
// File: sort3Numbers.cpp
// reads three numbers and sorts them in ascending order
#include <iostream>
using namespace std;

// Functions used
// Sorts a pair of numbers

void order(float& , float&) ;        // INOUT – number to sort

int main ()
{
    float num1 ,num2 , num3 ;  // user input – numbers to sort

    // Read 3 numbers
    cout << "Enter 3 numbers to sort: " ;
    cin >> num1 >> num2 >> num3 ;

    // Sort them.
```

```
    order(num1 , num2);                    // order data in num1 & num2
    order(num1 , num3);                    // order data in num1 & num3
    order(num2 , num3) ;                   // order data in num2 & num3

    // Display results.
    cout << "The three numbers in order are: " << endl ;
    cout << num1 << " " << num2 << " " << num3 << endl ;

    return 0 ;
}

// Sorts a pair of numbers represented by x and y
// Pre: x and y are assigned values.
// Post: x is the smaller of the pair and y is the larger.
void order(float& x , float& y)    // INOUT – numbers to sort
{
    // Local data
    float temp ;

    // Compare x and y , exchange values if not in order.
    if (x > y)
    {                              // exchange the values in x and y
        temp = x;                  // store old x in temp
        x = y ;                    // store old y in x
        y = temp ;                 // store old x in y
    }
} // end order
```

ทดสอบ

```
Enter 3 numbers to sort:  7.5  9.6  5.5
The three numbers in order are :
5.5  7.5  9.6
```

4.3 คำสั่ง switch

คำสั่ง switch เป็นคำสั่งที่ใช้สำหรับตรวจสอบค่าของข้อมูลที่มีค่าเดียว แตกต่างจากคำสั่ง if ที่การตรวจสอบมักเป็นช่วงของข้อมูล ที่เราค่อนข้างคุ้นเคย อาทิเช่นการเลือกการทำงานจากเมนู -- ซึ่งผู้ใช้เลือกเพียงเมนูเดียว โดยข้อมูลในการเลือกอาจเป็นตัวเลข หรือ ตัวอักษร 1 ตัว หรือข้อมูลชนิด bool ก็ได้ แต่ใช้กับข้อมูลที่เป็น string หรือ float ไม่ได้

switch Statement

รูปแบบ:

```
switch (selector)
{
   case label 1  : statements 1 ;
                   break ;
   case label 2  : statements 2 ;
                   break ;
   . . .
   case label n : statements n ;
                   break ;
   default :  statements d ; // optional
}
```

การทำงานของคำสั่ง switch นั้นขึ้นอยู่กับ selector โดยค่าของ selector อาจเป็นตัวแปร หรือเป็นนิพจน์คำสั่ง ก็ได้ ซึ่งอาจเป็นตัวเลขจำนวนเต็ม หรือ ตัวอักษร 1 ตัว หรือเป็นข้อมูลชนิด bool ก็ได้ ซึ่งมีค่าได้เพียง 1 ค่า โดยการทำงานจะนำค่า selector มาเปรียบเทียบกับ กรณีต่างๆ ว่าตรงกับกรณีใด ก็จะทำงานตามคำสั่งจนกระทั่งพบคำสั่ง break จะหยุดการทำงานและหลุดจากคำสั่ง switch เพื่อกระทำคำสั่งต่อไป แต่ถ้าการตรวจสอบค่านั้นไม่เท่ากับกรณีใดเลย จะกระทำคำสั่งหลัง default ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 4.8 การเลือกโน๊ตเพลง

```
// Display a musical note
switch (musicalNote)
{
    case 'c' :    cout << "do" ;
                  break ;
    case 'd' :    cout << "re" ;
                  break ;
    case 'e' :    cout << "mi" ;
                  break ;
    case 'f' :    cout << "fa" ;
                  break ;
    case 'g' :    cout << "sol" ;
                  break ;
    case 'a' ;    cout << "la" ;
                  break ;
  . case 'b' :    cout << "ti" ;
                  break ;
    default :     cout << "An invalid note was read. " << endl ;
}
```

สำหรับตัวอย่างนี้ musicalNote เป็นตัวแปรชนิด char การทำงานของคำสั่ง switch จะตรวจสอบว่าค่าของตัวแปรว่าตรงกับกรณีใด ในกรณีที่ตรงกับกรณีใดจะพิมพ์โน๊ตเพลงนั้นออกทางจอภาพแต่ถ้าไม่ตรงกับกรณีใดเลยจะพิมพ์ว่า "An invalid note was read"

ตัวอย่างที่ 4.9 เปรียบเทียบคำสั่ง if และ คำสั่ง switch

ในการแก้ปัญหาโปรแกรมนั้น โปรแกรมเมอร์อาจเลือกใช้คำสั่ง if หรือคำสั่ง switch ในการแก้ปัญหาก็ได้ ซึ่งผลลัพธ์ของการทำงานเหมือนกัน เช่น

```
if ((momOrDad == 'M') || (momOrDad == 'm'))
    cout << "Hello Mom – Happy Mother's Day" << endl ;
else
    cout << "Hello Dad – Happy Father's Day" << endl ;
```

การทำงานของคำสั่ง if นี้มีทางเลือก 2 ทางคือตรวจสอบว่าตัวแปร momOrDad มีค่าเป็น 'M' หรือ 'm' หรือ ตัวอักษรตัวอื่น การตรวจสอบตัวอักษร1 ตัวนั้นต้องตรวจสอบทั้งอักษรตัวใหญ่และอักษรตัวเล็กเพราะภาษา C++ ถือว่าเป็นตัวอักษรคนละตัวกัน เราสามารถแปลงคำสั่ง if เป็นคำสั่ง switch ได้ดังนี้

```
switch (momOrDad)
{
   case 'M' : case 'm' :
      cout << "Hello Mom – Happy Mother's Day " << endl ;
      break ;
   case 'D' : case 'd' :
      cout << "Hello Dad – Happy Father's Day " << endl ;
      break ;
}
```

การทำงานของคำสั่ง switch นั้นในกรณีที่การทำงานอาจมีค่าของ selector ได้หลายๆค่า เราจะเขียน case หรือกรณีที่เป็นทางเลือกต่อเนื่องกันเป็นลำดับ ซึ่งมีค่าใดที่เท่ากับ selector ก็จะกระทำคำสั่งตามทางเลือกนั้นๆจนกระทั่งพบคำสั่ง break

ตัวอย่างที่ 4.10 ฟังก์ชันการแสดงเกรด

จากตัวอย่างที่ 4.4 ซึ่งเป็นการแสดงเกรดจากช่วงของคะแนนต่างๆเราสามารถเขียนให้อยู่ในรูปแบบของคำสั่ง switch ได้ดังนี้

```
void displayGrade (int score)
{
    switch (score/10 )
    {
     case 9 : case 10 :
                cout << "Grade is A "
                     << endl ;
                break ;
    case 8 :
                cout << "Grade is B "
                     << endl ;
                break ;
    case 7 :
                cout << "Grade is C "
                     << endl ;
                break ;
    case 6 :
```

```
                cout << "Grade is D "
                     << endl ;
                break ;
    default :
                cout << "Grade is F "
                     << endl ;
    }
}
```

การทำงานของฟังก์ชันนี้ เมื่อเรียกใช้ displayGrade(75); จะส่งค่า 75 ให้แก่ตัวแปร score ในฟังก์ชัน ดังนั้น selector ในที่นี้เป็นนิพจน์ทางคณิตศาสตร์ จะนำ 75 / 10 มีค่าเท่ากับ 7 ดังนั้น การทำงานจะเลือกกระทำในกรณีที่เป็น case 7 โดยจะพิมพ์ "Grade is C " ออกทางจอภาพ สิ้นสุดการทำงานของคำสั่ง switch เมื่อพบคำสั่ง break ; และจบการทำงานของฟังก์ชัน

สรุป

1. โปรแกรมหนึ่งๆนั้นมีโครงสร้างการปฏิบัติการอยู่ 3 รูปแบบได้แก่ sequence , selection และ repetition
2. นิพจน์ทางตรรก จะให้ผลลัพธ์เป็น true หรือ false อย่างใดอย่างหนึ่งเท่านั้น
3. ตัวกระทำทางตรรก || (or) จะให้ผลลัพธ์จากการเปรียบเทียบเท่ากับ true ถ้านิพจน์ใดนิพจน์หนึ่งหรือทั้งสองนิพจน์มีค่าเท่ากับ true
4. ตัวกระทำทางตรรก && จะให้ผลลัพธ์จากการเปรียบเทียบเท่ากับ true ถ้าทั้งสองนิพจน์มีค่าเท่ากับ true เท่านั้น
5. ตัวกระทำ ! คือคอมพลีเมนต์ (complement) หรือไม่(negative) จะให้ผลลัพธ์ตรงข้าม
6. ภาษา C++ นำค่า ASCII CODE ที่แทนตัวอักษรต่างๆนั้นมาใช้สำหรับเปรียบเทียบในกรณีที่เป็นข้อความจะเปรียบเทียบตัวอักษรทีละตัวจากซ้ายไปขวา
7. เราสามารถนำค่าของนิพจน์ทางตรรกให้แก่ตัวแปรชนิด bool ได้ ตามรูปแบบดังนี้

   variable = expression ;
8. if statement with a dependent statement เป็นคำสั่งที่มีทางเลือกในการทำงาน 2 ทางแต่จะกระทำตามคำสั่งเมื่อเงื่อนไขเป็น true เท่านั้นแต่ถ้าเป็น false จะไม่มีการทำงานใดๆ มีรูปแบบดังนี้

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

   ในกรณีที่ทางเลือกของการปฏิบัติงานนั้นมีการทำงานหลายๆคำสั่ง(Compound statements) ต้องเรียนคำสั่งเหล่านั้นภายใต้เครื่องหมาย { } ดังนี้

```
if (condition)
{
    compound statement ;
}
```

9. if statement with two alternatives เป็นคำสั่งที่มีทางเลือกในการทำงาน 2 ทาง โดยถ้าตรวจสอบเงื่อนไขเป็น true จะทำคำสั่งทางหนึ่ง และถ้าเงื่อนไขเป็น false จะกระทำตาม

คำสั่งอีกทางหนึ่ง มีรูปแบบดังนี้

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

10. Multiple-Alternative Decision เป็นคำสั่งที่มีทางเลือกหลายทางโดยเลือกการทำงานทางใดทางหนึ่งเท่านั้นที่มีเงื่อนไขเป็นจริง มีรูปแบบดังนี้

```
if (condition 1)
    statement 1 ;
else if (condition 2)
    statement 2 ;
. . .
else if (condition n)
    statement n ;
```

11. คำสั่ง switch เป็นคำสั่งที่ใช้สำหรับตรวจสอบค่าของข้อมูลที่มีค่าเดี่ยว แตกต่างจากคำสั่ง if ที่การตรวจสอบมักเป็นช่วงของข้อมูล มีรูปแบบดังนี้

```
switch (selector)
    {
    case label 1 : statements 1 ;
                   break ;
    . . .
    case label n : statements n ;
                   break ;
    default : statements d ; // optional
    }
```

แบบฝึกหัด

1. กำหนดให้ x = 10.0 และ y = 15.0 จงหาค่าของเงื่อนไขต่อไปนี้

   1.1 x != y

   1.2 x < y

   1.3 x >= (y –x)

   1.4 x == (y + x – y)

2. กำหนดให้ a = 5 , b =10 , c = 15 และ flag = false จงหาค่าของเงื่อนไขต่อไปนี้

   2.1 (c == (a * b)) || !flag

   2.2 (a != 7) && flag || ((a + c) <= 20)

   2.3 !(b <= 12) && (a%2 == 0)

   2.4 !((a > 5) || (c < (a+b)))

3. พิจารณาว่านิพจน์ในข้อใดต่อไปนี้ไม่ถูกต้อง จงให้เหตุผลด้วยว่าทำไม
   กำหนดให้ x , y เป็นข้อมูลชนิด float และ p ,q, r เป็นข้อมูลชนิด bool

   3.1 x < 5.1 && y > 22.3

   3.2 p && q || q && r

4. พิจารณาว่าคำสั่งต่อไปนี้แสดงผลอย่างไรทางจอภาพ

   4.1
```
if (12 < 12)
    cout << "less" << endl ;
else
    cout << "not less" <, endl ;
```

   4.2
```
var1 = 15.0 ;
var2 = 25.12 ;
if (2 * var1 >= var2)
        cout << "O.K." << endl;
else
        cout << "Not O.K." << endl ;
```

5. จงหาค่าของตัวแปร X จากส่วนของคำสั่งต่อไปนี้ ถ้ากำหนดให้ค่า y = 15.0

5.1
```
x = 25.0 ;
if (y != (x – 10.0 ))
        x = x – 10.0 ;
else
        x = x / 2.0 ;
```

5.2
```
if ( y < 15.0) && ( y >= 0.0 )
        x = 5 * y ;
else
        x = 2 * y ;
```

6. จงเขียนฟังก์ชันในการแสดงสารสนเทศจากคะแนนเฉลี่ยที่ได้จากตารางต่อไปนี้

Grade point Average	Transcript Message
0.0 to 0.99	Failed semester – registration suspended
1.0 to 1.99	On probation for next semester
2.0 to 2.99	(no message)
3.0 to 3.49	Dean's list for semester
3.5 to 4.0	Highest honors for semester

7. จงแปลงคำสั่ง switch ต่อไปนี้ให้เป็นคำสั่ง if

```
switch (x>y)
{
  case 1 :
      cout << "x greater" << endl ;
      break ;
  case 0 :
      cout << "y greater or equal" << endl ;
      break ;
}
```

8. จงเขียนคำสั่ง switch ในการพิมพ์ข้อความ "red" ทางจอภาพถ้าค่าของตัวแปร color มีค่าเท่ากับ 'r' หรือ 'R' พิมพ์ข้อความ "yellow" ถ้าค่าของตัวแปร color มีค่าเท่ากับ 'y' หรือ 'Y' พิมพ์ข้อความ "blue" ถ้าค่าของตัวแปร color มีค่าเท่ากับ 'b' หรือ 'B'
9. จงเขียนฟังก์ชันในการหาค่าผลต่างของ x และ y ถ้าค่าของตัวแปร x มากกว่า y จะได้ผลลัพธ์เป็น x-y แต่ถ้าค่าของ y มีค่ามากกว่าค่า x จะส่งค่า y-x กลับมายังจุดเรียกใช้
10. จงเขียนฟังก์ชันในการหาค่าสัมบูรณ์ (absolute value) ของเลขจำนวนจริงใดๆ
11. จงเขียนฟังก์ชันในการวาดรูป วงกลม(circle) , สี่เหลี่ยม(square) , สามเหลี่ยม(Triangle) และเขียนโปรแกรมในการอ่านอักขระ C , S หรือ T เพื่อเลือกการวาดรูปที่ต้องการ
12. จงเขียนโปรแกรมในการหาอายุของบุคคลใดๆ โดยการป้อนวัน เดือน ปีเกิด และวัน เดือน ปีปัจจุบัน เพื่อหาอายุของบุคคลคนนั้น
13. จงเขียนโปรแกรมเพื่อหาส่วนลดในการซื้อสินค้าของบริษัทแห่งหนึ่ง โดยมีการแสดงสารสนเทศติดต่อกับผู้ใช้รวมทั้งเงื่อนไขที่ได้รับสิทธิประโยชน์แตกต่างกันดังนี้ดังนี้

	ยอดเงินซื้อสินค้า	
ประเภทของลูกค้า	1000.00-49999.99	>50000.00
(1) สมาชิกปีแรก	3%	5%
(2) สมาชิกบัตรเงิน	5%	10%
(3) สมาชิกบัตรทอง	10%	15%
(4) ไม่เป็นสมาชิก	0%	3%

โดยโปรแกรมจะมีสารสนเทศเพื่อบอกให้ผู้ใช้ป้อนประเภทของลูกค้า และยอดเงินซื้อสินค้าเพื่อหาส่วนลดที่ลูกค้าได้รับ โปรแกรมจะคำนวณส่วนลดที่ลูกค้าได้รับเป็นเงินบาท และยอดสุทธิที่ต้องจ่ายหลังจากหักส่วนลดแล้วออกทางจอภาพ

14. จงเขียนโปรแกรมรับปี คศ.ใดๆทางแป้นพิมพ์จงหาว่าเป็นปีอธิกสุรธินหรือไม่ ในกรณีที่เป็นจริงจะพิมพ์ว่า "366 days" แต่ถ้าไม่ใช่พิมพ์ "365 days " โดยปีอธิกสุรธินเป็นปีที่มีทั้งหมด 366 วัน
15. จงเขียนโปรแกรมอ่านตัวเลขใดๆ 4 จำนวน จงหาจำนวนที่มากที่สุด และจำนวนที่น้อยที่สุดพิมพ์ออกมาทางจอภาพ