หน่วยที่ 2 ตัวดำเนินการและนิพจน์ (Operators and Expressions)

ตัวดำเนินการ (Operator)

ตัวดำเนินการ จะเป็นตัวช่วยให้ผู้ใช้งานสามารถดำเนินการจัดการกับข้อมูลหรือตัวแปรที่สร้างขึ้นได้ตามที่ต้องการ โดยหลักการใช้งานตัวดำเนินการของภาษาซีจะคล้ายกับหลักการใช้งานตัวดำเนินการของคณิตศาสตร์ อย่างเช่น เครื่องหมายคำนวณต่าง ๆ การเปรียบเทียบค่าความจริงและเท็จของนิพจน์ เป็นต้น แต่หากจะมีรายละเอียดการใช้งานที่แตกต่างกันบ้างเล็กน้อย ซึ่งในการเขียนโปรแกรมองค์ประกอบที่สำคัญสิ่งหนึ่ง คือ ตัวดำเนินการ ซึ่งมักจะนำไปใช้ร่วมกับนิพจน์ในลักษณะต่าง ๆ ตัวดำเนินการมีหลายประเภทดังต่อไปนี้

1. ตัวดำเนินการกำหนดค่า (Assignment Operator)
2. ตัวดำเนินการคณิตศาสตร์ (Arithmetic Operator)
3. ตัวดำเนินการกำหนดค่าเชิงประกอบ (Compound Assignment Operator)
4. ตัวดำเนินการเอกภาค (Unary Operator)
5. ตัวดำเนินการเปลี่ยนชนิดข้อมูล (Type Cast Operator)
6. ตัวดำเนินการความสัมพันธ์ (Relational Operator)
7. ตัวดำเนินการความเท่ากัน (Equality Operator)
8. ตัวดำเนินการตรรกะ (Logical Operator)
9. ตัวดำเนินการเงื่อนไข (Conditional Operator)
10. ตัวดำเนินการคอมม่า (Comma Operator)
11. ตัวดำเนินการระดับบิต (Bitwise operator)

1. ตัวดำเนินการกำหนดค่า (Assignment Operator)

ตัวดำเนินการกำหนดค่าเป็นตัวดำเนินการพื้นฐานที่ใช้สำหรับสั่งให้คอมพิวเตอร์คำนวณ หรืออาจใช้ในการกำหนดค่าเริ่มต้นให้กับตัวแปร หรือย้ายค่าตัวแปรจากที่หนึ่งไปยังอีกที่หนึ่งก็ได้ โดยใช้เครื่องหมาย **เท่ากับ (=)**

ความหมายของตัวดำเนินการกำหนดค่า คือ การกำหนดค่าที่อยู่ทางขวามือของคำสั่งให้กับตัวแปรที่อยู่ทางซ้ายมือ หรือ การกำหนดค่าที่อยู่ทางขวามือของเครื่องหมายเท่ากับ (=) ให้กับตัวแปรที่อยู่ทางซ้ายมือ ดังนี้

รูปแบบ

ตัวแปร = ตัวแปร ;	ตัวแปร = ค่าคงที่ ;	ตัวแปร = นิพจน์ ;

ความหมาย

เครื่องหมาย = (เท่ากับ) มีความหมายแตกต่างจากคณิตศาสตร์
ในภาษาซี หมายถึง การกำหนดค่า (Assignment) ฝั่งขวา ให้กับ ตัวแปรฝั่งซ้าย โดย

ตัวแปร คือ การนำค่าตัวแปรฝั่งขวาไปเก็บในตัวแปรฝั่งซ้าย

ค่าคงที่ คือ การนำค่าคงที่ไปเก็บในตัวแปรฝั่งซ้าย

นิพจน์ คือ การนำค่าผลลัพธ์การประมวลผลจากนิพจน์ ไปเก็บในตัวแปรฝั่งซ้าย

ตัวอย่าง เช่น

ตัวแปร	ค่าคงที่	นิพจน์
i = j ; x = y ; a = b ;	age = 10 ; j = 3.55 ; ch = 'A' ;	speed = distance / time ; total = 60 * hours + minutes ; pi = 22.0 / 7.0 ;

นอกจากนี้ยังมีรูปแบบเพิ่มเติม หากต้องการกำหนดค่าให้กับตัวแปรหลายตัวด้วยค่าเดียวกัน มีรูปแบบดังนี้

รูปแบบ

```
ตัวแปร1 = ตัวแปร2 = ... = นิพจน์ ;
```

โดยตัวแปรทั้งหมดจะต้องเป็นตัวแปรชนิดเดียวกัน เช่น

```
int   x , y , z ;
x = y = z = 50 ;
```

แปลความหมายได้ว่า

```
z = 50 ;
y = z ;
x = y ;
```

นอกจากนี้ เมื่อมีการใช้ตัวดำเนินการกำหนดค่า จะมีการแปลงค่าชนิดของข้อมูลโดยอัตโนมัติอีกด้วย (Implicit Casting) ซึ่งจะเกิดกับข้อมูลตัวเลข เช่น กำหนดค่าที่มีชนิดเป็นจำนวนเต็ม int ให้กับตัวแปรที่มีชนิดเป็นจำนวนจริง float จะเกิดกระบวนการแปลงค่า จาก int ไปเป็น float เช่น

```
float  x ;
x = 14 ;
```

14 เป็นข้อมูลชนิดจำนวนเต็ม int แต่ x เป็นตัวแปรประเภทจำนวนจริง float ฉะนั้นค่า 14 จะถูกแปลงเป็น 14.000000 ก่อนที่จะนำไปเก็บยังตัวแปร x

แต่หากมีการกำหนดค่าที่มีชนิดเป็นจำนวนจริง float ให้กับตัวแปรที่มีชนิดเป็นจำนวนเต็ม int จะเกิดการปัดเศษที่เป็นทศนิยมทิ้ง ตัวอย่างเช่น

```
int  a ;
a = 3.5 ;
```

จะได้ว่า มีการแปลงค่า 3.5 ไปเป็น 3 ก่อนที่จะนำค่านั้นไปกำหนดให้กับตัวแปร a จะได้ว่า a เก็บค่า 3 พิจารณาตัวอย่างต่อไปนี้

```
int      a ;
float    x ;
x = a = 12.74 ;
```

จะได้ว่า มีการกำหนดค่า a = 12.74 ก่อน ซึ่งมีการแปลงค่า 12.74 โดยปัดเศษทศนิยมทิ้ง ได้ว่า ตัวแปร a เก็บค่า 12 จากนั้นนำค่าใน a ไปกำหนดค่าให้กับตัวแปร x ซึ่งเป็นจำนวนจริง float จะมีการแปลงค่า 12 เป็น 12.000000 จะได้ว่า ตัวแปร x เก็บค่า 12.000000

2. ตัวดำเนินการคณิตศาสตร์

ในการคำนวณทางคณิตศาสตร์ของภาษาซี มีการใช้เครื่องหมายสำหรับการคำนวณ เรียกว่า **โอเปอเรเตอร์** (Operator) มีรายละเอียดดังต่อไปนี้

โอเปอเรเตอร์	ความหมาย	ตัวอย่าง	ผลลัพธ์
+	การบวก (Addition)	3 + 4 5.8 + 3 c = a + b ;	7 8.8
–	การลบ (Subtraction)	5 – 3 1.5 – 1 c = a – b ;	2 0.5
*	การคูณ (Multiplication)	6 * 5 2.5 * 2 c = a * b ;	30 5.0
/	การหาร (Division) ถ้าตัวตั้งและตัวหารต่างเป็นจำนวนเต็ม ผลลัพธ์จะเป็นจำนวนเต็ม แต่ถ้าตัวตั้งหรือตัวหารตัวใดตัวหนึ่งเป็นจำนวนจริงที่มีทศนิยม ผลลัพธ์จะเป็นจำนวนจริงที่มีทศนิยมด้วย	10 / 5 6.0 / 3 1 / 2 1.0 / 2 22 / 7 22.0 / 7.0 c = a / b ;	2 2.0 0 0.5 3 3.14159265...
%	หารเอาเศษ (Modulus) ที่เป็นจำนวนเต็ม	5 % 3 9 % 4 c = a % b ;	2 1

ข้อควรระวัง การหาร (Division) ผลลัพธ์ที่ได้จะขึ้นอยู่กับประเภทชนิดข้อมูลของตัวตั้งและตัวหารด้วย

จำนวนเต็ม (Integer)	/	จำนวนเต็ม (Integer)	=	จำนวนเต็ม (Integer) ตัดเศษทศนิยมทิ้ง
จำนวนเต็ม (Integer)	/	จำนวนจริง (Float)	=	จำนวนจริง (Float)
จำนวนจริง (Float)	/	จำนวนเต็ม (Integer)	=	จำนวนจริง (Float)
จำนวนจริง (Float)	/	จำนวนจริง (Float)	=	จำนวนจริง (Float)

3. ตัวดำเนินการกำหนดค่าเชิงประกอบ (Compound Assignment Operator)

ตัวดำเนินการกำหนดค่าเชิงประกอบ เป็นตัวดำเนินการที่ผสมระหว่างตัวดำเนินการกำหนดค่าและตัวดำเนินการทางคณิตศาสตร์ มีดังนี้

โอเปอเรเตอร์	ความหมาย	ตัวอย่าง	คำสั่งเต็ม
+=	การบวกเชิงประกอบ	a += 2 ;	a = a + 2 ;
–=	การลบเชิงประกอบ	a –= 2 ;	a = a – 2 ;
*=	การคูณเชิงประกอบ	a *= 2 ;	a = a * 2 ;
/=	การหารเชิงประกอบ	a /= 2 ;	a = a / 2 ;
%=	หารเอาเศษเชิงประกอบ	a %= 2 ;	a = a % 2 ;

ตัวอย่างที่ 1

กำหนดให้ a = 5 และ b = 10

พิจารณาการดำเนินการต่อไปนี้

a *= 10 ; ผลลัพธ์ a มีค่าเท่ากับ 50 (มีค่าเท่ากับ a = a * 10)

b += 5 ; ผลลัพธ์ b มีค่าเท่ากับ 15 (มีค่าเท่ากับ b = b + 5)

ตัวอย่างที่ 2

กำหนดให้ i = 2 และ j = 10

พิจารณาการดำเนินการต่อไปนี้

j *= 3 + 2 ; ผลลัพธ์ j มีค่าเท่ากับ 50 เท่ากับคำสั่ง j = j * (3 + 2)

j += j – i ; ผลลัพธ์ j มีค่าเท่ากับ 18 เท่ากับคำสั่ง j = j + (j – i)

การแปลคำสั่งของตัวดำเนินการกำหนดค่าเชิงประกอบ จะทำคำสั่งที่อยู่ทางขวามือก่อนเสมอ เนื่องจากตัวดำเนินการกำหนดค่าเชิงประกอบมีลำดับความสำคัญต่ำกว่าตัวดำเนินการอื่น ๆ

ตัวอย่างที่ 3 หลังจากเรียนเรื่องตัวดำเนินการเอกภาค (Unary Operator)

กำหนดให้ i = 2 และ j = 10

พิจารณาการดำเนินการต่อไปนี้

j += ++j – --i ; ผลลัพธ์ j มีค่าเท่ากับ 21

เท่ากับคำสั่ง j = j + (++j – --i) ; หรือ j = 11 + (11 – 1)

j += ++j – i-- ; ผลลัพธ์ j มีค่าเท่ากับ 20

เท่ากับคำสั่ง j = j + (++j – i--) ; หรือ j = 11 + (11 – 2)

4. ตัวดำเนินการเอกภาค (Unary Operator)

ตัวดำเนินการเอกภาค คือ การใช้ตัวดำเนินการกับตัวแปรตัวเดียว ในที่นี้จะแสดงการใช้ตัวดำเนินการ 2 ตัว กับตัวแปรตัวเดียว ซึ่งเป็นการเขียนคำสั่งเพื่อเพิ่มค่าหรือลดค่าตัวแปรชนิดจำนวนเต็มทีละ 1 ค่าในแบบย่อ โดยอาศัยตัวดำเนินการเอกภาคเพิ่มค่า (++) และตัวดำเนินการเอกภาคลดค่า (––) มีลักษณะการใช้งาน 2 แบบ คือ

1. ตัวดำเนินการเอกภาคเติมหน้า (Prefix mode)

เช่น ++a ; มีค่าเท่ากับ a = a + 1 คือ การเพิ่มค่าครั้งละหนึ่งค่า

––a ; มีค่าเท่ากับ a = a – 1 คือ การลดค่าครั้งละหนึ่งค่า

2. ตัวดำเนินการเอกภาคเติมเหลัง (Postfix mode)

เช่น a++ ; มีค่าเท่ากับ a = a + 1 คือ การเพิ่มค่าครั้งละหนึ่งค่า

a–– ; มีค่าเท่ากับ a = a – 1 คือ การลดค่าครั้งละหนึ่งค่า

โอเปอเรเตอร์	ความหมาย	ตัวอย่าง	ผลลัพธ์
++	**การเพิ่มค่าทีละ 1 (Increment)** ++a จะเพิ่มค่าของ a ขึ้น 1 ก่อน แล้วจึงนำค่าของ a ไปใช้	a = 10 ; b = ++a ; มีความหมายเช่นเดียวกับ a = a + 1 ; b = a ;	b = 11 , a = 11
	a++ จะนำค่าของ a ไปใช้ก่อน แล้วจึงเพิ่มค่าของ a ขึ้น 1	a = 10 ; b = a++ ; มีความหมายเช่นเดียวกับ b = a ; a = a + 1 ;	b = 10 , a = 11
––	**ลดค่าที่ละ 1 (Decrement)** ––a จะลดค่าของ a ลง 1 ก่อน แล้วจึงนำค่าของ a ไปใช้	a = 10 ; b = ––a ; มีความหมายเช่นเดียวกับ a = a – 1 ; b = a ;	b = 9 , a = 9
	a–– จะนำค่าของ a ไปใช้ก่อน แล้วจึงลดค่าของ a ลง 1	a = 10 ; b = a–– ; มีความหมายเช่นเดียวกับ b = a ; a = a – 1 ;	b = 10 , a = 9

ตัวอย่างที่ 1

กำหนดให้ x = 5

พิจารณาการดำเนินการต่อไปนี้

z = ++x ; ผลลัพธ์ z = 6 และ x = 6 (เพิ่มค่า x ก่อนแล้วนำค่าไปเก็บที่ตัวแปร z)

z = x++ ; ผลลัพธ์ z = 5 และ x = 6 (เพิ่มค่า x หลังจากนำค่าไปเก็บที่ตัวแปร z)

z = --x ; ผลลัพธ์ z = 4 และ x = 4 (ลดค่า x ก่อนแล้วนำค่าไปเก็บที่ตัวแปร z)

z = x-- ; ผลลัพธ์ z = 5 และ x = 4 (ลดค่า x หลังจากนำค่าไปเก็บที่ตัวแปร z)

ตัวอย่างที่ 2

กำหนดให้ a = 20

พิจารณาการดำเนินการต่อไปนี้

b = a++ * 2 ; ผลลัพธ์ b = 40 และ a = 21 (เพิ่มค่า a หลังจากการคูณ)

c = ++a * 2 ; ผลลัพธ์ c = 42 และ a = 21 (เพิ่มค่า a ก่อนแล้วนำไปคูณ)

ตัวอย่างที่ 3

กำหนดให้ i = 10 และ j = 2

พิจารณาการดำเนินการต่อไปนี้

i * j++ ; ผลลัพธ์จะเท่ากับ 20 และ j = 3 (เพิ่มค่า j หลังจากการคูณ)

i * ++j ; ผลลัพธ์จะเท่ากับ 30 และ j = 3 (เพิ่มค่า j ก่อนจากการคูณ)

i * j-- ; ผลลัพธ์จะเท่ากับ 20 และ j = 1 (ลดค่า j หลังจากการคูณ)

i * --j ; ผลลัพธ์จะเท่ากับ 10 และ j = 1 (ลดค่า j ก่อนจากการคูณ)

5. ตัวดำเนินการเปลี่ยนชนิดข้อมูล (Type Cast Operator)

การเปลี่ยนชนิดข้อมูลในภาษาซี มี 2 ลักษณะ คือ การเปลี่ยนชนิดข้อมูลโดยไม่ตองใช้คำสั่ง (Implicit Casting) ซึ่งกลาวถึงแล้วในหัวขอที่ 1 และการเปลี่ยนชนิดข้อมูลโดยใช้คำสั่ง (Explicit Casting) เชน หากตองแปลงข้อมูลชนิดจำนวนจริง float ไปเป็นข้อมูลชนิดจำนวนเต็ม int จะตองใช้คำสั่ง

```
int  a ;
a = (int)12.423 ;
```

```
int   a ;
float  b ;
b = 12 ;
b = (float)a :
```

จะได้ว่า a มีคาเทากับ 12 กระบวนการทำงานจะมีการเปลี่ยนชนิดข้อมูลที่อยู่ใกลกับตัวดำเนินการเปลี่ยนชนิดข้อมูลให้เป็นชนิดข้อมูลที่ระบุในวงเล็บ () แลวจึงมีการกำหนดค่าใหม่นั้นใหกับ a ทั้งนี้สามารถกำหนดชนิดข้อมูลที่จะเปลี่ยนค่าเป็นชนิดข้อมูลใด ๆ ก็ได้ แสดงตัวอย่างเพิ่มเติมดังตัวอย่าง เช่น

```
int  x;
x = 2.5 * 2 ;          ผลลัพธ์ x จะมีค่าเท่ากับ 5
x = (int)2.5 * 2 ;     ผลลัพธ์ x จะมีค่าเท่ากับ 4
x = (int)(2.5 * 2) ;   ผลลัพธ์ x จะมีค่าเท่ากับ 5
```

6. ตัวดำเนินการความสัมพันธ์ (Relational Operator)

ตัวดำเนินการความสัมพันธ์ เป็นตัวดำเนินการที่ใช้เปรียบเทียบนิพจน์ 2 นิพจน์ หรือนำข้อมูลสองค่ามาเปรียบเทียบกัน มักใช้กับคำสั่งควบคุมแบบทางเลือก ซึ่งจะกล่าวถึงในหน่วยต่อไป โดยข้อมูลทั้งสองค่าจะต้องเป็นข้อมูลประเภทเดียวกัน มีดังต่อไปนี้

ตัวดำเนินการ	ความหมาย	ตัวอย่าง	ค่าที่ได้ (ผลลัพธ์)
>	มากกว่า (greater than)	5 > 3	1 (True)
<	น้อยกว่า (less than)	8 < 2	0 (False)
>=	มากกว่าหรือเท่ากับ (equal & greater than)	10 >= 10	1 (True)
<=	น้อยกว่าหรือเท่ากับ (equal & less than)	10 <= 15	1 (True)

ผลลัพธ์ที่ได้จากการใช้ตัวดำเนินการความสัมพันธ์ คือ **จริง (True)** หรือ เท็จ **(False)** ซึ่งในภาษาซีแทนด้วยเลขจำนวนเต็ม กรณีที่เป็น**เท็จ** จะแทนด้วยค่า **0** และกรณีที่เป็น**จริง** จะแทนด้วยค่า **1** หรือค่าที่**ไม่ใช่ 0**

ตัวอย่าง การประมวลผลนิพจน์จากตัวดำเนินการความสัมพันธ์ กำหนดให้ a = 5 และ b = 3

นิพจน์	แปลงนิพจน์	ค่าตรรกะ	ค่าที่ได้
5 + 2 * 4 **<** (5 + 2) * 4	5 + 2 * 4 **<** (5 + 2) * 4	True	1
a + b **<=** b + a	(5 + 3) **<=** (3 + 5)	True	1
a / b **<** b / a	(5 / 3) **<** (3 / 5)	False	0

7. ตัวดำเนินการความเท่ากัน (Equality Operator)

ตัวดำเนินการความเท่ากัน เป็นตัวดำเนินการที่ใช้เปรียบเทียบความเท่ากันของนิพจน์ 2 นิพจน์ หรือนำข้อมูลสองค่ามาเปรียบเทียบความเท่ากัน มักใช้กับคำสั่งควบคุมแบบทางเลือก ซึ่งจะกล่าวถึงในหน่วยต่อไป มีดังต่อไปนี้

ตัวดำเนินการ	ความหมาย	ตัวอย่าง	ค่าที่ได้ (ผลลัพธ์)
==	เท่ากับ (equal)	5 == 10	0 (False)
!=	ไม่เท่ากับ (not equal)	5 != 5	0 (False)

ผลลัพธ์ของการเปรียบเทียบที่ได้จากการใช้ตัวดำเนินความเท่ากัน คือ **จริง (True)** หรือ เท็จ **(False)** ซึ่งการใช้งานจะต้องระวังเพราะมีความสับสนระหว่างการใช้ตัวดำเนินการความเท่ากัน เครื่องหมายเท่ากับ **==** กับตัวดำเนินการกำหนดค่า **=** ซึ่งมีการทำงานที่ต่างกัน และตัวดำเนินการความเท่ากัน เครื่องหมายไม่เท่ากับ **!=** ไม่ใช่เครื่องหมาย **<>** เหมือนในโปรแกรมภาษาอื่น ๆ เช่น

a == 2 เป็นการเปรียบเทียบว่า ตัวแปร a มีค่าเท่ากับ 2 หรือไม่ (จริงหรือเท็จ)

a = 2 เป็นการกำหนดค่า 2 ให้กับตัวแปร a

ในการเปรียบเทียบค่าจะต้องระมัดระวังเรื่องของเลขทศนิยมซึ่งเกิดจากการคำนวณของเครื่องคอมพิวเตอร์ ซึ่งคอมพิวเตอร์แต่ละเครื่องอาจจะให้ผลการคำนวณที่ต่างกัน ดังตัวอย่าง

```
float    a = 1.0 ;
```

คำสั่ง a == a / 3.0 * 3.0 ; อาจจะให้ค่าที่คาดไม่ถึง เนื่องจากในทางคณิตศาสตร์ 1 / 3 * 3 จะได้เท่ากับ 1 แต่ในทางคอมพิวเตอร์ เครื่องคอมพิวเตอร์บางเครื่อง เมื่อนำ 1.0 / 3.0 จะได้ค่า 0.33333... เมื่อนำมาคูณกับ 3.0 จะได้ค่า 0.999999... ซึ่งผลลัพธ์จะไม่เท่ากับ 1.0 ตามที่ผู้เขียนโปรแกรมตั้งใจ

ตัวอย่าง การประมวลผลนิพจน์จากตัวดำเนินการความเท่ากัน กำหนดให้ a = 5 และ b = 3

นิพจน์	แปลงนิพจน์	ค่าตรรกะ	ค่าที่ได้
7 == 4	7 == 4	False	0
a != b	5 != 3	True	1
a + b == b + a	(5 + 3) == (3 + 5)	True	1

8. ตัวดำเนินการตรรกะ (Logical Operator)

ตัวดำเนินการตรรกะ เป็นตัวดำเนินการที่ใช้คู่กับตัวดำเนินการความสัมพันธ์และตัวดำเนินการความเท่ากัน ซึ่งมักจะใช้กับคำสั่งควบคุมแบบทางเลือก ซึ่งจะกล่าวถึงในหน่วยต่อไป การทำงานเหมือนกับการเปรียบเทียบเชิงตรรกะทั่วไป คือ ใช้แนวคิดเชิงเหตุผลที่เกี่ยวข้องกับค่า 2 ค่า คือ จริง (True) แทนค่าด้วย 1 และเท็จ (False) แทนค่าด้วย 0 ดังนี้

ตัวดำเนินการ	ความหมาย	การดำเนินการ
&&	และ (Logical **AND**)	ผลลัพธ์ของเงื่อนไขจะมีค่าเป็น**จริง** เมื่อเงื่อนไขทั้งหมดเป็นจริง
||	หรือ (Logical **OR**)	ผลลัพธ์ของเงื่อนไขจะมีค่าเป็น**เท็จ** เมื่อเงื่อนไขทั้งหมดเป็นเท็จ
!	นิเสธ (Logical **NOT**)	จะเปลี่ยนค่าของเงื่อนไขเป็น**ตรงข้าม** จากจริงเป็นเท็จ และจากเท็จเป็นจริง

ตารางแสดงค่าความจริงของตัวดำเนินการตรรกะในกรณีต่าง ๆ ที่เป็นไปได้ทั้งหมดของ AND, OR, NOT โดยกำหนดให้ P และ Q คือ ประโยคตรรกะที่ให้ค่าจริง (True) หรือเท็จ (False)

P	Q	P && Q	P || Q	!P
T	T	**T**	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	**F**	T

หมายเหตุ T = True (จริง) F = False (เท็จ)

ตัวอย่างของการเปรียบเทียบ ได้แก่

```
!10          ผลลัพธ์   0   (False)
!0           ผลลัพธ์   1   (True)
!'A'         ผลลัพธ์   0   (False)
```

ตัวอย่าง การประมวลผลนิพจน์จากตัวดำเนินการตรรกะ กำหนดให้ int a = 1 , b = 3 , c = 4 ;

นิพจน์	แปลงนิพจน์	ค่าตรรกะ	ค่าที่ได้
a + b > 0 **&&** b / c > 1	((a + b) > 0) **&&** ((b / c) > 1)	False	0
a / c == b / c **&&** (a + b) / c == 1	((a / c) == (b / c)) **&&** (((a + b) / c) == 1)	True	1
!a **||** !b **||** !c	(!a) **||** (!b) **||** (!c)	False	0

ในการใช้ตัวดำเนินการตรรกะสิ่งที่ต้องระวัง คือ **Short Circuit** เนื่องจากทราบว่า ในกรณีของ **&&** (Logical **AND**) หากนิพจน์แรกเป็น**เท็จ** ไม่ว่านิพจน์ที่ 2 จะเป็นอะไร จะทำให้ผลลัพธ์ของนิพจน์นั้นเป็น**เท็จเสมอ** และในกรณีของ **||** (Logical **OR**) หากนิพจน์แรกเป็น**จริง** ไม่ว่าพจน์ที่ 2 จะเป็นอะไร จะทำให้ผลลัพธ์ของนิพจน์นั้นเป็น**จริงเสมอ** คอมไพเลอร์ใช้หลักการคิดนี้เช่นเดียวกัน ทำให้ไม่มีการประมวลผลในพจน์ที่ 2 ในกรณีที่พจน์แรกทำให้เกิด Short Circuit ดังตัวอย่าง

```
int     a = 10 ;                    //ประกาศตัวแปร a เป็นจำนวนเต็ม มีค่าเริ่มต้นเท่ากับ 10
printf("%d", a < 10 && a++ > 10);   //แสดงผลลัพธ์จาก a < 10 && a++ > 10  ได้ 0 (False)
printf("\n%d", a);                  //แสดงผลลัพธ์ค่าที่เก็บในตัวแปร a  จะแสดงค่า 10 ออกมา
```

จากตัวอย่าง ต้องการจะเพิ่มค่า a ขึ้น 1 หลังจากทำงานฟังก์ชัน printf() ในคำสั่งแรก แต่เนื่องจากนิพจน์ a < 10 เป็นเท็จ ทำให้นิพจน์หลังไม่ถูกประมวลผล จึงไม่เกิดการเพิ่มค่าให้ตัวแปร a จะได้ a มีค่าเท่ากับ 10 เหมือนเดิม

พิจารณาตัวอย่าง Short Circuit กรณีของการใช้ตัวดำเนินการ **||** (Logical OR)

```
int     a = 10 ;                    //ประกาศตัวแปร a เป็นจำนวนเต็ม มีค่าเริ่มต้นเท่ากับ 10
printf("%d", a < 20 || a++ > 10);   //แสดงผลลัพธ์จาก a < 20 && a++ > 10  ได้ 1 (True)
printf("\n%d", a);                  //แสดงผลลัพธ์ค่าที่เก็บในตัวแปร a  จะแสดงค่า 10 ออกมา
```

จากตัวอย่างนิพจน์ a < 20 เป็นจริง ทำใหเกิด Short Circuit คือ ไม่มีการประมวลผลในนิพจน์ที่ 2 ของตัวดำเนินการ || (Logical OR) คาตัวแปร a จึงไม่ถูกเพิ่มคาขึ้น จะได้คำตอบวา a มีคาเทากับ 10 เหมือนเดิม

นอกจากนี้การเขียนนิพจน์ทางตรรกศาสตร์ที่คุ้นเคยในชีวิตประจำวัน ในบางลักษณะไม่สามารถทำได้ในทางคอมพิวเตอร์ ตัวอย่างเช่น หากต้องการทราบว่า x มีค่าอยู่ในช่วงตั้งแต่ 10 ถึง 20 จริงหรือไม่ ปกติแล้วจะเขียนว่า 10 <= x <= 20 หรือในทางคณิตศาสตร์ $10 \leq x \leq 20$ ซึ่งรูปแบบนี้ไม่สามารถใช้งานได้ในทางคอมพิวเตอร์ ที่ถูกต้องจะเขียนว่า

```
x >= 10 && x <= 20
```

9. ตัวดำเนินการเงื่อนไข (Conditional Operator)

ตัวดำเนินการเงื่อนไขเป็นตัวดำเนินการที่ใช้ตรวจสอบเงื่อนไขของนิพจน์ มีรูปแบบดังนี้

รูปแบบ

นิพจน์1 **?** นิพจน์2 **:** นิพจน์3 **;**

ความหมาย

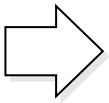
นิพจน์1 คือ **เงื่อนไข**ที่กำหนดขึ้น เพื่อใช้ตัดสินว่าจะทำงานหรือไม่ทำงานตามคำสั่ง ซึ่งเงื่อนไขอาจอยู่ในรูปของนิพจน์ (Expression) การคำนวณเปรียบเทียบ หรือเป็นคำสั่งของตัวแปรก็ได้ ที่จะมีการเปรียบเทียบเงื่อนไขให้ได้ผลลัพธ์ว่า **จริง** (True) หรือ**เท็จ** (False)

นิพจน์2 คือ คำสั่งที่จะให้ทำงาน ถ้าผลลัพธ์ของการตรวจสอบเงื่อนไข (Condition) ในส่วนของ**นิพจน์1** ออกมาเป็น**จริง** (True) หรือไม่ใช่ 0 (is non zero)

นิพจน์3 คือ คำสั่งที่จะให้ทำงาน ถ้าผลลัพธ์ของการตรวจสอบเงื่อนไข (Condition) ในส่วนของ**นิพจน์1** ออกมาเป็น**เท็จ** (False) หรือเท่ากับ 0 (is zero)

ซึ่งมีความหมายเดียวกับคำสั่งควบคุมทางเลือก ของการเลือกทำแบบสองเส้นทาง (if-else) ดังนี้

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



```
if (นิพจน์1)
        นิพจน์2 ;
else
        นิพจน์3 ;
```

คำอธิบาย condition คือ เงื่อนไขที่กำหนดขึ้น เพื่อใช้ตัดสินว่าจะทำงานหรือไม่ทำงานตามคำสั่ง โดยเงื่อนไขจะต้องเขียนอยู่ในเครื่องหมาย ( ) ซึ่งเงื่อนไขอาจอยู่ในรูปของนิพจน์ (Expression) การคำนวณเปรียบเทียบ หรือเป็นคำสั่งของตัวแปรก็ได้ ที่จะมีการเปรียบเทียบเงื่อนไขให้ได้ผลลัพธ์ว่า เป็น**จริง** (True) หรือเป็น**เท็จ** (False)

statement1 คือ คำสั่งที่จะให้ทำงาน ถ้าผลการตรวจสอบเงื่อนไข (Condition) ออกมาเป็น**จริง (True)** ซึ่งอาจจะมีคำสั่งที่ต้องการให้ทำงานมากกว่าหนึ่งคำสั่งก็ได้ แต่ต้องเขียนคำสั่งเหล่านั้นไว้ภายในเครื่องหมาย { }

statement2 คือ คำสั่งที่จะให้ทำงาน ถ้าผลการตรวจสอบเงื่อนไข (Condition) ออกมาเป็น**เท็จ (False)** ซึ่งอาจจะมีคำสั่งที่ต้องการให้ทำงานมากกว่าหนึ่งคำสั่งก็ได้ แต่ต้องเขียนคำสั่งเหล่านั้นไว้ภายในเครื่องหมาย { }

การประมวลผลจะตรวจสอบนิพจน์1 ว่าเป็นจริงหรือเท็จ หากเป็นจริงจะทำงานในนิพจน์2 แต่หากเป็นเท็จจะทำงานในนิพจน์3 ตัวอย่างเช่น

ตัวอย่าง การประมวลผลนิพจน์จากตัวดำเนินการเงื่อนไข กำหนดให้ int a = 10 , b = 20 ;

นิพจน์	แปลงนิพจน์	ค่าตรรกะ	ค่าที่ได้	ผลลัพธ์
c = (a > 10) ? a : b ;	c = (10 > 10) ? a : b ;	False	0	c = 20
c = ((a + b) > 10) ? a+10 : b+10 ;	c = ((10+20) > 10) ? a+10 : b+10 ;	True	1	c = 20
min = (a < b) ? a : b ;	min = (10 < 20) ? a : b ;	True	1	min = 10
max = (a > b) ? a : b ;	min = (10 > 20) ? a : b ;	False	0	max = 20

10. ตัวดำเนินการคอมม่า (Comma Operator)

ตัวดำเนินการคอมมา เป็นตัวดำเนินการเพื่อบอกถึงลำดับการทำงาน เพื่อช่วยกำหนดการทำงานของนิพจน์หนึ่ง ๆ มักจะใช้กับคำสั่งควบคุมแบบทางเลือก ซึ่งจะกล่าวถึงในหน่วยต่อไป มีรูปแบบดังนี้

รูปแบบ

```
นิพจน์1 , นิพจน์2 ;
```

การทำงานจะทำงานที่นิพจน์ทางซ้ายมือก่อนเสมอ แล้วจึงทำงานที่นิพจน์ขวามือ แต่ถ้ามใช้คำสั่ง

```
s = ( t = 2 , t + 3 ) ;
```

ลำดับการทำงาน คือ t = 2 หลังจากนั้นจะนำค่าใน t คือ 2 ไปบวกกับ 3 จะได้ค่า คือ 5 แล้วนำค่า 5 ไปกำหนดให้กับ s จะได้ว่า t มีค่าเท่ากับ 2 และ s มีค่าเท่ากับ 5

แต่หากไม่มีเครื่องหมายวงเล็บ ( ) ดังตัวอย่าง

```
s = t = 2 , t + 3 ;
```

จะได้ว่า s เท่ากับ 2 และ t เท่ากับ 2 หลังจากนั้นจะนำค่า t ไปบวกกับ 3 ได้ผลลัพธ์ของนิพจน์เป็น 5 มักจะใช้กับนิพจน์ตรรกศาสตร์ในคำสั่งควบคุมแบบทางเลือก ซึ่งจะกล่าวถึงในหน่วยต่อไป

11. ตัวดำเนินการระดับบิต (Bitwise operator)

ตัวดำเนินการที่ดำเนินการในระบิตของข้อมูล โดยทั่วไปแล้วตัวดำเนินการระดับบิตมักใช้ในการเขียนโปรแกรมระดับต่ำ (Low level) เนื่องจากมีการทำงานที่เรียบง่ายและรวดเร็วในการคำนวณและเปรียบเทียบค่า เพราะการทำงานนั้นเกิดขึ้นกับแต่ละบิตในหน่วยความจำโดยตรง ซึ่ง **บิต (bit)** ย่อมาจาก Binary digit คือ หน่วยข้อมูลที่เล็กที่สุดในคอมพิวเตอร์ มีค่าเป็น 0 หรือ 1

ตัวดำเนินการระดับบิตนั้นทำงานคล้ายกับตัวดำเนินการตรรกะ AND OR และ NOT แต่แทนที่จะจัดการกับค่าจริง (True) และเท็จ (False) มันจะเป็นการทำงานกับแต่ละบิต 1 และ 0 ของข้อมูลแทน การกระทำของตัวดำเนินการระดับบิตเป็นการกระทำที่เข้าถึงในระดับบิตของข้อมูล โดยที่ค่าของแต่ละบิตเป็นได้แค่ 1 หรือ 0 เท่านั้น มีดังนี้

ตัวดำเนินการ	ความหมาย	การดำเนินการ
&	Bitwise AND	ได้ผลลัพธ์เป็น 1 ถ้าเงื่อนไขค่าบิตทั้งสองเป็น 1 ไม่เช่นนั้นได้ผลลัพธ์เป็น 0
|	Bitwise OR	ได้ผลลัพธ์เป็น 0 ถ้าเงื่อนไขค่าบิตทั้งสองเป็น 0 ไม่เช่นนั้นได้ผลลัพธ์เป็น 1
^	Bitwise XOR (Exclusive-or)	ได้ผลลัพธ์เป็น 1 ถ้าค่าบิตทั้งสองแตกต่างกัน ไม่เช่นนั้นได้ผลลัพธ์เป็น 0
~	Bitwise NOT	กลับบิตจาก 1 เป็น 0 และกลับบิตจาก 0 เป็น 1
>>	เลื่อนบิตไปทางขวา (Shift Right)	เลื่อนบิตไปทางขวา n ตำแหน่ง บิตทางขวาสุด n ตำแหน่งจะถูกนำออกไป และเติมบิต 0 เข้ามาทางซ้าย
<<	เลื่อนบิตไปทางซ้าย (Shift Left)	เลื่อนบิตไปทางซ้าย n ตำแหน่ง และเติมบิต 0 เข้ามาทางขวาเป็นจำนวน n บิต

ตัวดำเนินการ	ความหมาย	ตัวอย่าง	คำสั่งเต็ม
&=	AND แล้วให้เท่ากับ	x &= y	x = x & y
|=	OR แล้วให้เท่ากับ	x |= y	x = x | y
^=	Exclusive-or แล้วให้เท่ากับ	x ^= y	x = x ^ y

ตารางแสดงค่าความจริงของตัวดำเนินการระดับบิต (Bitwise operators) ในกรณีต่าง ๆ ที่เป็นไปได้ทั้งหมดของ & , | , ^ และ ~ โดยกำหนดให้ P และ Q คือ ประโยคตรรกะที่ให้ค่าจริง (True) เท่ากับ 1 หรือเท็จ (False) เท่ากับ 0

P	Q	P & Q	P | Q	P ^ Q	~P
1	1	1	1	0	0
1	0	0	1	1	0
0	1	0	1	1	1
0	0	0	0	0	1

หมายเหตุ (**1** = T = True = จริง) (**0** = F = False = เท็จ)

ตัวอย่าง การทำงานของตัวดำเนินการระดับบิต

P	**1011 1010**	**1011 1010**	**1011 1010**	**1011 1010**
Q	**1101 1011**	**1111 1111**	**0000 0000**	**1111 0000**
P & Q	1001 1010	1011 1010	0000 0000	1011 0000
P | Q	1111 1011	1111 1111	1011 1010	1111 1010
P ^ Q	0110 0001	0100 0101	1011 1010	0100 1010
~P	0100 0101	0100 0101	0100 0101	0100 0101

ตัวอย่างที่ 1

```
int  x , y , result1 , result2 , result3 , result4 ;
x = 0x9C ;
y = 0x46 ;
```

จงหาผลลัพธ์ของ

1) result1 = x & y ;

2) result2 = x | y ;

3) result2 = x ^ y ;

4) result3 = ~x ;

วิธีคิด

เนื่องจากเป็นการกระทำแบบบิตต่อบิต จึงควรแปลงค่าจากเลขฐานสิบหกเป็นเลขฐานสอง ซึ่งก็คือ

x = 0x9C => 0000000010011100 (ตัวแปร int มีขนาดเป็น 16 บิต)

y = 0x46 => 0000000001000110 (ตัวแปร int มีขนาดเป็น 16 บิต)

1) result1 = (0000000010011100) & (0000000001000110)

0000000010011100

AND

0000000001000110

0000000000000100 => **0x0004 หรือ 0x04**

2) result2 = (0000000010011100) | (0000000001000110)

0000000010011100

OR

0000000001000110

0000000011011110 => **0x00DE หรือ 0xDE**

3) result3 = (0000000010011100) ^ (0000000001000110)

0000000010011100

XOR

0000000001000110

0000000011011010 => **0x00DA หรือ 0xDA**

4) result4 = ~(0000000010011100)

0000000010011100

NOT

1111111101100011 => **0xFF63**

การเลื่อนบิต

ในการเลื่อนบิต จะต้องบอกจำนวนครั้งการเลื่อนด้วยว่าให้มีการเลื่อนกี่ครั้ง มี 2 รูปแบบ

1. >> เลื่อนบิตไปทางขวา (Shift Right) ให้เลื่อนบิตไปทางขวา n ตำแหน่ง บิตทางขวาสุด n ตำแหน่งจะถูกนำออกไป และเติมบิต 0 เข้ามาทางซ้าย

2. << เลื่อนบิตไปทางซ้าย (Shift Left) เลื่อนบิตไปทางซ้าย n ตำแหน่ง และเติมบิต 0 เข้ามาทางขวาเป็นจำนวน n บิต

ตัวอย่างที่ 2

```
int     data , x1 , x2 , x3 ;
data = 0x93 ;
```

จงหาผลลัพธ์

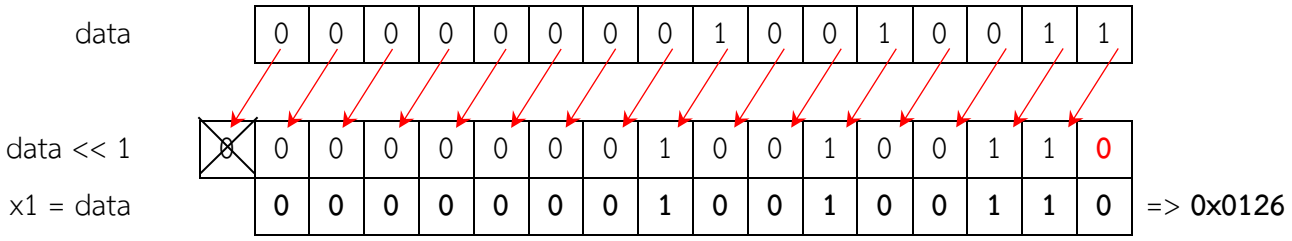
1) x1 = data << 1 ;
2) x2 = data << 2 ;
3) x3 = data >> 3 ;

วิธีคิด

เนื่องจากเป็นการกระทำแบบบิตต่อบิต จึงควรแปลงค่าจากเลขฐานสิบหกเป็นเลขฐานสองก่อนเลื่อนบิต ซึ่งก็คือ

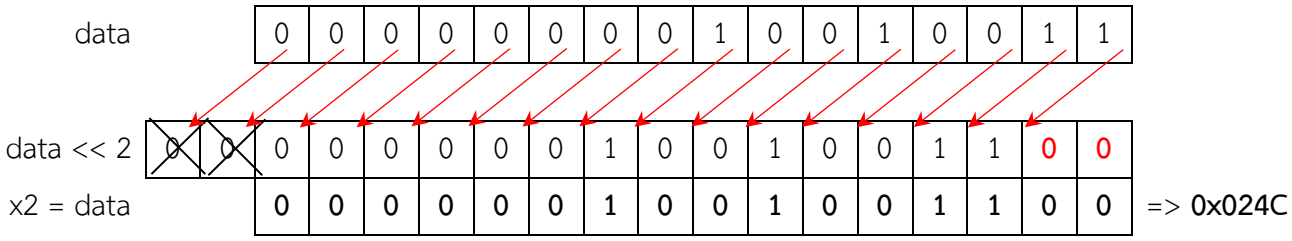
data = 0x93 => 0000000010010011 (ตัวแปร int มีขนาดเป็น 16 บิต)

1) x1 = data << 1 ;



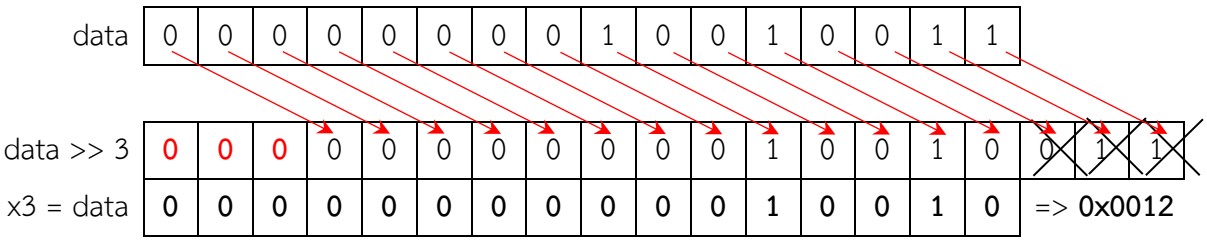
x1 = 0x0126 ตรงกับเลขฐานสิบ คือ 294

2) x2 = data << 2 ;



x2 = 0x024C ตรงกับเลขฐานสิบ คือ 588

3) x3 = data >> 3 ;



x3 = 0x0012 ตรงกับเลขฐานสิบ คือ 18

ตัวอย่างที่ 3

```
int  a , b , c ;
a = 0x7A ;
b = 0x16 ;
c = 0xFD ;
```

จงหาผลลัพธ์ของ

1) a &= 0x3C ;
2) b |= 0x51 ;
3) c ^= 0xD0 ;

วิธีคิด

1) a &= 0x3C ;

จาก a &= 0x3C ก็คือ a = a & 0x3C หมายความว่า นำค่าของ a (คือ 0x7A) ไป & (Bitwise AND) กับ 0x3C แล้วผลลัพธ์ที่ได้นำไปเก็บที่ a อีกครั้งหนึ่ง

เทียบได้เป็น a = (0000000001111010) & (0000000000111100)

0000000001111010

AND

0000000000111100

0000000000111000 => **0x0038 หรือ 0x38**

2) b |= 0x51 ;

จาก b |= 0x51 ก็คือ b = b | 0x51 หมายความว่า นำค่าของ b (คือ 0x16) ไป | (Bitwise OR) กับ 0x51 แล้วผลลัพธ์ที่ได้นำไปเก็บที่ b อีกครั้งหนึ่ง

เทียบได้เป็น b = (0000000000010110) & (0000000001010001)

0000000000010110

OR

0000000001010001

0000000001010111 => **0x0057 หรือ 0x57**

3) c ^= 0xD0 ;

จาก c ^= 0xD0 ก็คือ c = c ^ 0xD0 หมายความว่า นำค่าของ c (คือ 0xFD) ไป ^ (Exclusive-or) กับ 0xD0 แล้วผลลัพธ์ที่ได้นำไปเก็บที่ c อีกครั้งหนึ่ง

เทียบได้เป็น c = (0000000011111101) ^ (0000000011010000)

0000000011111101

XOR

0000000011010000

0000000000101101 => **0x002D หรือ 0x2D**

การเขียนนิพจน์ในภาษาซี

นิพจน์ (Expression) ในภาษาซี ได้แก่ ค่าคงตัว ค่าคงที่ หรือตัวแปร 1 จำนวน หรือกลุ่มของค่าคงตัว ค่าคงที่ หรือตัวแปรพร้อมด้วยตัวดำเนินการของภาษาซี หรือการนำข้อมูลและตัวแปรในภาษาซีมาดำเนินการด้วยเครื่องหมายทางคณิตศาสตร์ ตรรกศาสตร์ หรือเครื่องหมายเปรียบเทียบ ที่มีความหมายในภาษาซีเพื่อใช้เป็นตัวสั่งงาน

1. นิพจน์ทางคณิตศาสตร์

การเขียนนิพจน์ทางคณิตศาสตร์ในภาษาซีจะเหมือนกับการเขียนนิพจน์ทางคณิตศาสตร์ตามปกติ เพียงแต่เปลี่ยนมาใช้เครื่องหมายทางคณิตศาสตร์ของภาษาซีแทน

นิพจน์ทางคณิตศาสตร์ตามปกติ	นิพจน์ทางคณิตศาสตร์ในภาษาซี
$X^2 + 4X + 2$	X * X + 4 * X + 2
$5X^2 + Y$	5 * X * X + Y
2ab + 4c	2 * a * b + 4 * c
$\frac{K^2}{KM+2}$	(K * K) / (K * M + 2)
$\frac{x^2}{xy+2}$	(x * x) / (x * y + 2)
$\frac{a+b}{c-d}$	(a * b) / (c – d)
4z	4 * z

2. นิพจน์ทางตรรกศาสตร์

ตัวอย่างการเขียนนิพจน์ทางตรรกศาสตร์ พร้อมทั้งผลลัพธ์จากการดำเนินการทางตรรกศาสตร์ แสดงได้ดังตารางต่อไปนี้ โดยกำหนดให้ ตัวแปร a = 25 , b = -124 และ c = 0

การเขียนนิพจน์ทางตรรกศาสตร์	การดำเนินการ	ผลลัพธ์
c && (a <= b)	F && F	F
(b >= c) || (!a)	F || F	F
(a > b) && (c <= a)	T && T	T
(!a) || (a == b)	F || F	F
(a + 5) > (b – 100)	30 > -224	T

ลำดับความสำคัญของเครื่องหมาย

ลำดับความสำคัญ	เครื่องหมาย	หมายเหตุ
1	( )	ถ้าเครื่องหมายมีลำดับการดำเนินการระดับเดียวกันให้ดำเนินการจาก**ซ้ายไปขวา**
2	! , ++ , –– , (type cast)	
3	* , / , %	
4	+ , –	
5	< , <= , > , >=	
6	== , !=	
7	&&	
8	||	
9	+= , –= , *= , /= , %=	

ตัวอย่างที่ 1

จากนิพจน์ทางคณิตศาสตร์ 17 % 5 * 3 ผลลัพธ์ที่ได้จากนิพจน์นี้ คือ

จากนิพจน์ลำดับของการกระทำ คือ

❶ นำ 17 % 5 ก่อน = 2 * 3

❷ นำผลที่ได้คูณ 3 = 6

ผลลัพธ์ที่ได้จากนิพจน์นี้ คือ 6

ตัวอย่างที่ 2

จากนิพจน์ทางคณิตศาสตร์ 3 + 5 * 10 – 2 ผลลัพธ์ที่ได้จากนิพจน์นี้ คือ

จากนิพจน์ลำดับของการกระทำ คือ

❶ นำ 5 คูณด้วย 10 = 3 + 50 – 2

❷ นำ 3 บวกกับผลที่ได้จากการคูณ = 53 – 2

❸ นำผลที่ได้ลบด้วย 2 = 51

ผลลัพธ์ที่ได้จากนิพจน์นี้ คือ 51

ตัวอย่างที่ 3

จากนิพจน์ทางคณิตศาสตร์ (3 + 5) * 10 – 2 ผลลัพธ์ที่ได้จากนิพจน์นี้ คือ

จากนิพจน์ลำดับของการกระทำ คือ

❶ ทำในวงเล็บก่อน = 8 * 10 – 2

❷ นำผลที่ได้จากการทำในวงเล็บคูณด้วย 10 = 80 – 2

❸ นำผลที่ได้ลบด้วย 2 = 78

ผลลัพธ์ที่ได้จากนิพจน์นี้ คือ 78

ตัวอย่างที่ 4

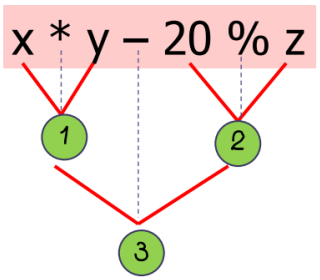
จากนิพจน์ทางคณิตศาสตร์ x * y – 20 % z ผลลัพธ์ที่ได้จากนิพจน์นี้ คือ

จากนิพจน์ลำดับของการกระทำ คือ

❶ x * y

❷ 20 % z

❸ ❶ – ❷



ตัวอย่างที่ 5

จากนิพจน์ทางคณิตศาสตร์ (a – b) * 10 / c && d + 5 ผลลัพธ์ที่ได้จากนิพจน์นี้ คือ

จากนิพจน์ลำดับของการกระทำ คือ

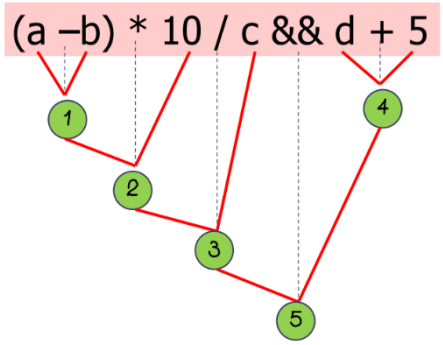
❶ a – b

❷ ❶ * 10

❸ ❷ / c

❹ d + 5

❺ ❸ && ❹



ตัวอย่างที่ 6

จากนิพจน์ (3 > 5) && (2 < 10) || (4 < 0) ผลลัพธ์ที่ได้จากนิพจน์นี้ คือ

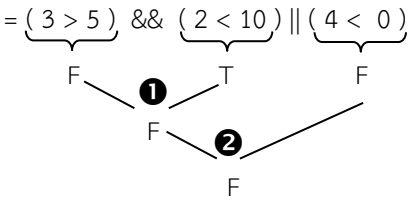
ลำดับของการกระทำของนิพจน์นี้ คือ

(3 > 5) && (2 < 10) || (4 < 0) = False && True || False

❶ ❷ = False || False

= False

ผลลัพธ์ที่ได้จากนิพจน์นี้ คือ **False (เท็จ)**



ตัวอย่างที่ 7

กำหนดให้ A = 1 , B = 2 , C = 3

จากนิพจน์ (A != B) && ((A > C) || (C > B)) ผลลัพธ์ที่ได้จากนิพจน์นี้ คือ

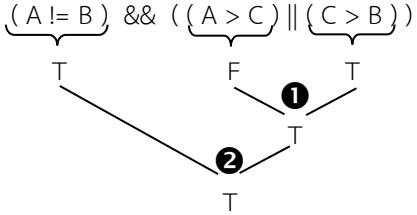
ลำดับของการกระทำของนิพจน์นี้ คือ

(A != B) && ((A > C) || (C > B)) = True && (False || True)

❷ ❶ = True && True

= True

ผลลัพธ์ที่ได้จากนิพจน์นี้ คือ **True (จริง)**



คอมพิวเตอร์กับเลขฐาน

ระบบเลขฐานที่มีความเกี่ยวข้องกับคอมพิวเตอร์ คือ ช่วยในเรื่องการจัดการระบบดิจิตอลหรือระบบอิเล็กทรอนิกส์ในคอมพิวเตอร์ หรือแทนรหัสข้อมูลในระบบ BCD, EBCDIC, ASCII โดยส่วนใหญ่ระบบเลขฐานที่ใช้ในคอมพิวเตอร์เป็นระบบเลขฐานสอง ระบบเลขฐานแปดและระบบเลขฐานสิบหก โดยจะต้องมีการนำระบบเลขฐานดังกล่าวมาคำนวณผลด้วยตัวดำเนินการทางคณิตศาสตร์ ตลอดจนกระทั่งการเปลี่ยนระบบเลขฐาน เพื่อให้มนุษย์เกิดความเข้าใจระบบการทำงานของคอมพิวเตอร์ซึ่งในการประมวลผลข้อมูลด้วยคอมพิวเตอร์นั้นข้อมูลต่าง ๆ จะถูกนำเข้าเป็นลำดับของบิต (Bit) หรือเลขฐานสองก่อน เช่น 110100110110 , 110101100110 , 110110110110

ระบบเลขฐาน

ระบบเลขฐานประกอบด้วยเลขฐาน 2 เลขฐาน 8 เลขฐาน 10 เลขฐาน 16

ระบบเลขฐาน 2 (Binary Number System) เป็นเลขฐานที่ประกอบด้วยเลข 2 ตัว ได้แก่ เลข 0 กับ เลข 1 ซึ่งเป็นเลขฐานที่คอมพิวเตอร์สามารถเข้าใจได้ง่าย เพราะว่าอุปกรณ์ทางไฟฟ้าก็มีสถานะเพียง 2 สถานะ คือ เปิด กับ ปิด ซึ่งก็เทียบได้กับ 0 กับ 1 แต่ถ้าใช้เลขฐาน 10 ในคอมพิวเตอร์อาจจะเกิดปัญหาอย่างอื่นตามมาหรือแม้แต่อุปกรณ์ทางไฟฟ้า ก็ต้องแบ่งสถานะออกเป็น 10 สถานะ ซึ่งไม่เป็นที่นิยมนัก การเก็บข้อมูลในระบบของคอมพิวเตอร์ก็จะจัดเก็บเป็นกลุ่มตัวเลขฐานสองหลายบิต ขึ้นอยู่กับขนาดของสิ่งที่ต้องการเก็บ และหน่วยความจำที่ใช้

ระบบเลขฐาน 8 (Octal Number System) เป็นเลขฐานที่ประกอบด้วยเลข 8 ตัว ซึ่งประกอบด้วยเลข 0, 1, 2, 3, 4, 5, 6, 7 ซึ่งเป็นเลขฐานที่เพิ่มเนื้อที่หน่วยความจำในการเก็บให้มากขึ้น การเก็บข้อมูลเป็นเลขฐาน 8 จะทำให้เก็บข้อมูลได้มากขึ้น

ระบบเลขฐาน 10 (Decimal Number System) เป็นเลขฐานที่ประกอบด้วยตัวเลข 10 ตัว ซึ่งประกอบด้วยเลข 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ซึ่งระบบเลขฐาน 10 เป็นระบบเลขฐานที่คนทั่วไปสามารถเข้าใจได้เป็นอย่างดี เพราะเป็นตัวเลขที่เกี่ยวข้องกับชีวิตประจำวันซึ่งใช้มาตลอด สามารถจำได้และคำนวณได้ง่ายกว่าเลขฐานอื่น ๆ

ระบบเลขฐาน 16 (Hexadecimal Number System) เป็นเลขฐานที่ประกอบด้วยตัวเลข 10 ตัวและตัวอักษรแทนตัวเลขอีก 6 ตัว ซึ่งประกอบด้วยเลข 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 และตัวอักษรภาษาอังกฤษแทน 10 ถึง 15 ได้แก่ A, B, C, D, E, F ซึ่งก็จะเก็บข้อมูลได้มากกว่าระบบเลขฐาน 2 และฐาน 8

ระบบเลขฐาน 2	0	1														
ระบบเลขฐาน 8	0	1	2	3	4	5	6	7								
ระบบเลขฐาน 10	0	1	2	3	4	5	6	7	8	9						
ระบบเลขฐาน 16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
											10	11	12	13	14	15

ตารางการแปลงเลขระหว่างระบบเลขฐานต่าง ๆ

เลขฐาน 2	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
เลขฐาน 8	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
เลขฐาน 10	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
เลขฐาน 16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

การแปลงค่าเลขฐาน

1. การแปลงเลขฐาน 10 เป็น เลขฐานใด ๆ ($?_{10} = ?_2$, $?_{10} = ?_8$, $?_{10} = ?_{16}$)
2. การแปลงเลขฐานใด ๆ เป็น เลขฐาน 10 ($?_2 = ?_{10}$, $?_8 = ?_{10}$, $?_{16} = ?_{10}$)
3. การแปลงเลขฐาน 2 เป็น เลขฐาน 8 หรือเลขฐาน 16 ($?_2 = ?_8$, $?_2 = ?_{16}$)
4. การแปลงเลขฐาน 8 เลขฐาน 16 เป็น เลขฐาน 2 ($?_8 = ?_2$, $?_{16} = ?_2$)
5. การแปลงเลขฐาน 8 เป็น เลขฐาน 16 ($?_8 = ?_{16}$)
6. การแปลงเลขฐาน 16 เป็น เลขฐาน 8 ($?_{16} = ?_8$)

1. การแปลงเลขฐาน 10 เป็น เลขฐานใด ๆ ($?_{10} = ?_2$, $?_{10} = ?_8$, $?_{10} = ?_{16}$)

ใช้การหารสั้น แล้วตอบเรียงเลขเศษจากล่างขึ้นบน เช่น

$58_{10} = ?_2$

2) 58
2) 29 เศษ 0
2) 14 เศษ 1
2) 7 เศษ 0
2) 3 เศษ 1
1 เศษ 1

∴ จะได้ $58_{10} = 111010_2$

$58_{10} = ?_8$

8) 58
7 เศษ 2

∴ จะได้ $58_{10} = 72_8$

$58_{10} = ?_{16}$

16) 58
3 เศษ 10

∴ จะได้ $58_{10} = 3A_{16}$

2. การแปลงเลขฐานใด ๆ เป็น เลขฐาน 10 ($?_2 = ?_{10}$, $?_8 = ?_{10}$, $?_{16} = ?_{10}$)

ให้คูณกับเลขฐานแล้วยกกำลัง 0, 1, 2, 3, ... แล้วหาผลรวม เช่น

$111010_2 = ?_{10}$

$111010_2 = (1x2^5) + (1x2^4) + (1x2^3) + (0x2^2) + (1x2^1) + (0x2^0)$
$= (1x32) + (1x16) + (1x8) + (0x4) + (1x2) + (0x1)$
$= 32 + 16 + 8 + 0 + 2 + 0$
$= 58$

∴ จะได้ $111010_2 = 58_{10}$

$2567_8 = ?_{10}$

$2567_8 = (2x8^3) + (5x8^2) + (6x8^1) + (7x8^0)$
$= (2x512) + (5x64) + (6x8) + (7x1)$
$= 1024 + 320 + 48 + 7$
$= 1399$

∴ จะได้ $2567_8 = 1399_{10}$

$A10_{16} = ?_{10}$

$A10_{16} = (10x16^2) + (1x16^1) + (0x16^0)$
$= (10x256) + (1x16) + (0x1)$
$= 2560 + 16 + 0$
$= 2576$

∴ จะได้ $A10_{16} = 2576_{10}$

3. การแปลงเลขฐาน 2 เป็น เลขฐาน 8 หรือเลขฐาน 16 ($?_2 = ?_8$, $?_2 = ?_{16}$)

3.1 การแปลงเลขฐาน 2 เป็น เลขฐาน 8 ($?_2 = ?_8$)

สามารถคำนวณได้จากการแบ่งกลุ่มเลขฐานสอง เป็นกลุ่มละสามหลัก จากด้านขวาไปด้านซ้าย แล้วแปลงเลขฐานสองแต่ละกลุ่มให้เป็นเลขฐานสิบ จากนั้นจึงนำตัวเลขที่ได้มาเรียงต่อกัน ซึ่งการแปลงเลขฐานสองให้เป็นเลขฐานสิบนั้นสามารถคำนวณได้จากข้อ 2 หรือเทียบจากตารางเลขฐาน เช่น

$111110101_2 = ?_8$

111110101_2 = 111 110 101 (แบ่งกลุ่มเป็น 3 หลัก จากด้านขวาไปด้านซ้าย)

= 7 6 5 (นำเลขฐานสองแต่ละกลุ่มมาแปลงให้เป็นเลขฐานสิบ)

= 765_8 (นำเลขที่ได้มาเรียงติดกันเป็นเลขฐานแปด)

∴ จะได้ $111110101_2 = 765_8$

$11101111_2 = ?_8$

11101111_2 = 011 101 111 (แบ่งกลุ่มเป็น 3 หลัก จากด้านขวาไปด้านซ้าย)

= 3 5 7 (นำเลขฐานสองแต่ละกลุ่มมาแปลงให้เป็นเลขฐานสิบ)

= 357_8 (นำเลขที่ได้มาเรียงติดกันเป็นเลขฐานแปด)

∴ จะได้ $11101111_2 = 357_8$

3.2 การแปลงเลขฐาน 2 เป็น เลขฐาน 16 ($?_2 = ?_{16}$)

สามารถคำนวณได้จากการแบ่งกลุ่มเลขฐานสอง เป็นกลุ่มละสี่หลัก จากด้านขวาไปด้านซ้าย แล้วแปลงเลขฐานสองแต่ละกลุ่มให้เป็นเลขฐานสิบ จากนั้นจึงนำตัวเลขที่ได้มาเรียงต่อกัน ซึ่งการแปลงเลขฐานสองให้เป็นเลขฐานสิบนั้นสามารถคำนวณได้จากข้อ 2 หรือเทียบจากตารางเลขฐาน เช่น

$111110101_2 = ?_{16}$

111110101_2 = 0001 1111 0101 (แบ่งกลุ่มเป็น 4 หลัก จากด้านขวาไปด้านซ้าย)

= 1 15 5 (นำเลขฐานสองแต่ละกลุ่มมาแปลงให้เป็นเลขฐานสิบ)

= $1F5_{16}$ (นำเลขที่ได้มาเรียงติดกันเป็นเลขฐานสิบหก)

∴ จะได้ $111110101_2 = 1F5_{16}$

$11101111_2 = ?_{16}$

11101111_2 = 1110 1111 (แบ่งกลุ่มเป็น 4 หลัก จากด้านขวาไปด้านซ้าย)

= 14 15 (นำเลขฐานสองแต่ละกลุ่มมาแปลงให้เป็นเลขฐานสิบ)

= EF_{16} (นำเลขที่ได้มาเรียงติดกันเป็นเลขฐานสิบหก)

∴ จะได้ $11101111_2 = EF_{16}$

4. การแปลงเลขฐาน 8 เลขฐาน 16 เป็น เลขฐาน 2 ($?_8 = ?_2$, $?_{16} = ?_2$)

4.1 การแปลงเลขฐาน 8 เป็น เลขฐาน 2 ($?_8 = ?_2$)

สามารถคำนวณได้จากการแบ่งเลขฐานแปดทีละหลัก แปลงเลขฐานแปดให้เป็นเลขฐานสองสามหลักด้วยการเปรียบเทียบจากตารางเลขฐาน หากเลขฐานสองนั้นมีไม่ถึงสามหลัก ให้เติม 0 ด้านหน้าของหลักนั้น แล้วจึงนำค่าที่ได้มาเรียงต่อกัน เช่น

$375_8 = ?_2$

375_8 = 3 7 5 (แบ่งกลุ่มเป็น 1 หลัก)

= 011 111 101 (แปลงเลขฐานแปดแต่ละหลัก ให้เป็นเลขฐานสอง 3 หลัก)

= 011111101_2 (นำเลขที่ได้มาเรียงติดกันเป็นเลขฐานสอง)

∴ จะได้ $375_8 = 11111101_2$

$164_8 = ?_2$

164_8 = 1 6 4 (แบ่งกลุ่มเป็น 1 หลัก)

= 001 110 100 (แปลงเลขฐานแปดแต่ละหลัก ให้เป็นเลขฐานสอง 3 หลัก)

= 001110100_2 (นำเลขที่ได้มาเรียงติดกันเป็นเลขฐานสอง)

∴ จะได้ $164_8 = 1110100_2$

4.2 การแปลงเลขฐาน 16 เป็น เลขฐาน 2 ($?_{16} = ?_2$)

สามารถคำนวณได้จากการแบ่งเลขฐานสิบหกทีละหลัก แปลงเลขฐานสิบหกให้เป็นเลขฐานสองสี่หลักด้วยการเปรียบเทียบจากตารางเลขฐาน หากเลขฐานสองนั้นมีไม่ถึงสี่หลัก ให้เติม 0 ด้านหน้าของหลักนั้น แล้วจึงนำค่าที่ได้มาเรียงต่อกัน เช่น

$2DF8_{16} = ?_2$

$2DF8_{16}$ = 2 D F 8 (แบ่งกลุ่มเป็น 1 หลัก)

= 0010 1101 1111 1000 (แปลงเลขฐานสิบหกแต่ละหลัก ให้เป็นเลขฐานสอง 4 หลัก)

= 0010110111111000_2 (นำเลขที่ได้มาเรียงติดกันเป็นเลขฐานสอง)

∴ จะได้ $2DF8_{16} = 10110111111000_2$

$A71_{16} = ?_2$

$A71_{16}$ = A 7 1 (แบ่งกลุ่มเป็น 1 หลัก)

= 1010 0111 0001 (แปลงเลขฐานสิบหกแต่ละหลัก ให้เป็นเลขฐานสอง 4 หลัก)

= 101001110001_2 (นำเลขที่ได้มาเรียงติดกันเป็นเลขฐานสอง)

∴ จะได้ $A71_{16} = 101001110001_2$

5. การแปลงเลขฐาน 8 เป็น เลขฐาน 16 ($?_8 = ?_{16}$)

สามารถคำนวณได้จากการแบ่งเลขฐานแปดทีละหลัก แล้วแปลงเลขฐานแปดให้เป็นเลขฐานสองด้วยการเปรียบเทียบจากตารางเลขฐาน แล้วนำเลขฐานสองที่ได้แปลงให้เป็นเลขฐานสิบหกอีกครั้งหนึ่ง เช่น

$765_8 = ?_{16}$

765_8 = 7 6 5 (แบ่งกลุ่มเป็น 1 หลัก)
= 111 110 101 (แปลงเลขฐานแปดแต่ละหลัก ให้เป็นเลขฐานสอง 3 หลัก)
= 111110101_2 (นำเลขที่ได้มาเรียงติดกันเป็นเลขฐานสอง)
= 0001 1111 0101 (แบ่งกลุ่มเป็น 4 หลัก จากด้านขวาไปด้านซ้าย)
= 1 15 5 (นำเลขฐานสองแต่ละกลุ่มมาแปลงให้เป็นเลขฐานสิบ)
= $1F5_{16}$ (นำเลขที่ได้มาเรียงติดกันเป็นเลขฐานสิบหก)

∴ จะได้ $765_8 = 1F5_{16}$

$234_8 = ?_{16}$

234_8 = 2 3 4 (แบ่งกลุ่มเป็น 1 หลัก)
= 010 011 100 (แปลงเลขฐานแปดแต่ละหลัก ให้เป็นเลขฐานสอง 3 หลัก)
= 010011100_2 (นำเลขที่ได้มาเรียงติดกันเป็นเลขฐานสอง)
= 0000 1001 1100 (แบ่งกลุ่มเป็น 4 หลัก จากด้านขวาไปด้านซ้าย)
= 0 9 12 (นำเลขฐานสองแต่ละกลุ่มมาแปลงให้เป็นเลขฐานสิบ)
= $9C_{16}$ (นำเลขที่ได้มาเรียงติดกันเป็นเลขฐานสิบหก)

∴ จะได้ $234_8 = 9C_{16}$

6. การแปลงเลขฐาน 16 เป็น เลขฐาน 8 ($?_{16} = ?_8$)

สามารถคำนวณได้จากการแปลงเลขฐานสิบหกทีละหลัก แล้วแปลงเลขฐานสิบหกให้เป็นเลขฐานสองด้วยการเปรียบเทียบจากตารางเลขฐาน แล้วนำเลขฐานสองที่ได้แปลงให้เป็นเลขฐานสิบหกอีกครั้งหนึ่ง เช่น

$1F5_{16} = ?_8$

$1F5_{16}$ = 1 F 5 (แบ่งกลุ่มเป็น 1 หลัก)
= 0001 1111 0101 (แปลงเลขฐานสิบหกแต่ละหลัก ให้เป็นเลขฐานสอง 4 หลัก)
= 000111110101_2 (นำเลขที่ได้มาเรียงติดกันเป็นเลขฐานสอง)
= 000 111 110 101 (แบ่งกลุ่มเป็น 3 หลัก จากด้านขวาไปด้านซ้าย)
= 0 7 6 5 (นำเลขฐานสองแต่ละกลุ่มมาแปลงให้เป็นเลขฐานสิบ)
= 0765_8 (นำเลขที่ได้มาเรียงติดกันเป็นเลขฐานแปด)

∴ จะได้ $1F5_{16} = 765_8$

ใบงานหน่วยที่ 2

1. จงบอกความหมายของตัวดำเนินการต่อไปนี้

ข้อที่	ตัวดำเนินการ	ความหมาย
1.1	+	
1.2	–	
1.3	*	
1.4	/	
1.5	%	
1.6	––	
1.7	++	
1.8	>	
1.9	>=	
1.10	<	
1.11	<=	
1.12	==	
1.13	!=	
1.14	&&	
1.15	||	
1.16	!	

2. จงหาผลลัพธ์ของการดำเนินการทางตรรกศาสตร์ต่อไปนี้

p	q	r	!q	p && r	q || r	(p && r) && (q || r)
T	T	T				
T	T	F				
T	F	T				
T	F	F				
F	T	T				
F	T	F				
F	F	T				
F	F	F				

3. จงหาผลลัพธ์ของการทำงานของตัวดำเนินการระดับบิตต่อไปนี้

x	1111 1111	1111 0000	1100 1100	0000 0000
y	1011 1010	1011 1010	1011 1010	1011 1010
x & y				
x | y				
x ^ y				
~x				

4. จงแปลงนิพจน์คณิตศาสตร์ต่อไปนี้ ให้เป็นนิพจน์ที่ใช้ในภาษาซี

นิพจน์ทางคณิตศาสตร์ตามปกติ	นิพจน์ทางคณิตศาสตร์ในภาษาซี
$5X^2 + 3X - 1$	
$X^2 + 5XY + Y^2$	
$2xy + 4y$	
πr^2	
$2\pi r$	
$\frac{x-y}{x+y}$	
$a = 2y^2 + y \times 3$	
$x = \frac{2y^2+4}{2}$	

5. จงหาผลลัพธ์ของนิพจน์ต่อไปนี้ พร้อมแสดงวิธีทำ

5.1 2 + 5 % 3

.....

.....

.....

.....

.....

5.2 2 * 3 + 4 * 2

.....

.....

.....

.....

.....

5.3 3 + 5 – 4 / 2

.....

.....

.....

.....

5.4 (8 > 5) || (3 < 2)

.....

.....

.....

.....

.....

5.5 (8 * 5) / 4 * 3

.....

.....

.....

.....

.....

5.6 1 + 4 * 6 + 8 / 4 + 4

.....

.....

.....

.....

6. ถ้า a = 5 , b = 3 และ c = 2 จงหาผลลัพธ์ของนิพจน์ต่อไปนี้

6.1 ++a * b++ _____

_____

6.2 --a + ++c _____

_____

8. หาที่ผิดในนิพจน์ตอไปนี้ และแก้ไขให้ถูกต้อง

8.1 a = + a – b * 32 _____

8.2 a = ab + c _____

8.3 x <> y _____

8.4 a => b _____

8.5 a1 > a2 OR a1 > a3 _____

8.6 10 <= x <= 20 _____

9. ใหเขียนนิพจน์ตามที่โจทย์ระบุ

9.1 มีตัวแปร a b และ c ใหเขียนนิพจน์เพื่อหาค่าของ a ซึ่งเกิดจาก b ยกกำลังสองบวกกับ c

_____

9.2 มีตัวแปร a b และ c ใหเขียนนิพจน์เพื่อหาค่าของ a ซึ่งเกิดจาก a คูณกับ c แลวหารด้วย b

_____

9.3 มีตัวแปร a b และ c ใหเขียนนิพจน์เพื่อเปรียบเทียบวา a บวก b มีคามากกวา a บวกด้งบ c หรือไม่

_____

9.4 มีตัวแปร a ใหเขียนนิพจน์เปรียบเทียบวา a มีคามากกวา 0 และเมื่อบวก a ด้วย 5 มีคาไม่เกิน 100

_____

9.5 มีตัวแปร x เก็บอายุของพนักงานคนหนึ่งในบริษัท ใหตรวจสอบวา x มีอายุอยู่ในชวงตั้งแต่ 25 ปขึ้นไป แต่วาไม่เกิน 60 ป

_____

9.6 มีตัวแปร x เก็บจำนวนจริง ใหตรวจสอบวา x มีค่าน้อยกว่า 100 หรือมากกว่า 500

_____

10. ใหคำตอบของนิพจน์ต่อไปนี้

10.1 int a=5 , b=10 , c = 15 , d ;

d = a + 5 * b; // d = _____

d = a + 5 * b; // d = _____

d = b / a * c; // d = _____

d = a + b * 3 / 2 – b / 4 + a % 2 + 10; // d = _____

d = 5

d += a + b % 3; // d = _____

d %= b - 3; // d = _____

```
d = a++ + b++ + 2;                          // d =_____
                                            // a =_____
                                            // b =_____
d = ++a + a++;                              // d =_____
                                            // a =_____
d = a + b++ 2 – b++ / 4 + ++a % 2;          // d =_____
                                            // a =_____
                                            // b =_____
```

10.2
```
int  a=5 , b ;
float  x = 12.5 , y ;
b = 15.7;                                   // b =_____
y = 10;                                     // y =_____
b = a / 2;                                  // b =_____
y = x / 2;                                  // y =_____
b = a % 3 * 2.5;                            // b =_____
y = x + 6 / 3;                              // y =_____
```

10.3
```
int  a = 5 , b = 10 , c = 3 ;
a > b || a < c                              // คำตอบ =_____
(a + b / 3) > 10                            // คำตอบ =_____
a*c > b && b*a/c < b/a*c                    // คำตอบ =_____
a >= 10 || b >=10 && c*a >= 10              // คำตอบ =_____
! (a < b) && (a > c) || !(a+c > b)          // คำตอบ =_____
```

11. เขียนโปรแกรมคำนวณพื้นที่ของวงกลม โดยรับรัศมีของวงกลมจากผู้ใช้ สูตรการหาพื้นที่ของวงกลม ได้แก่

$$CircleArea = \frac{1}{2} PI * r^2$$

กำหนดให้ค่า PI คือ 3.14159265

12. เขียนโปรแกรมรับข้อมูลจำนวนเต็ม 5 จำนวนจากผู้ใช้ และหาว่าค่าเฉลี่ยของข้อมูลที่รับเขามามีค่าเทาใด

13. เขียนโปรแกรมใหรับคาจำนวนจริงจากผู้ใช้ 1 จำนวน และใหหาวาเลขดังกลาวอยู่ในชวงของเลขจำนวนเต็มใด เชน หากผู้ใช้ป้อนเลข 12.5 ใหตอบวา “12.5 is between 12 and 13” (ใชตัวดำเนินการเปลี่ยนชนิดขอมูลในการเขียนโปรแกรม)

14. เขียนโปรแกรมเพื่อรับข้อมูลเลขจำนวนจริงจากผู้ใช้ 3 จำนวน ใหหาว่าค่าที่มากที่สุดที่ป้อนเข้ามาคือค่าใด โดยใช้ตัวดำเนินการเงื่อนไข