

บทที่ 4

ตัวดำเนินการและนิพจน์

(Operators and Expressions)

ในบทนี้ เราจะศึกษาถึงตัวดำเนินการในการแบบต่าง ๆ ที่ใช้ในภาษา C รวมทั้งการนำตัวดำเนินการไปใช้ในนิพจน์

4.1 การดำเนินการที่ใช้ในภาษา C จะประกอบด้วย

- 1) ตัวดำเนินการคำนวณ (arithmetic operators)
- 2) ตัวดำเนินการสัมพันธ์ (relational operators)
- 3) ตัวดำเนินการความเสมอภาค (equality operators)
- 4) ตัวดำเนินการตรรกยะ (logical operators)

ตารางที่ 4.1 แสดงตัวดำเนินการคำนวณ

ตัวดำเนินการ	จุดประสงค์
+	บวก
-	ลบ
*	คูณ
/	หาร
%	หาเศษที่ได้จากการหาร

ตารางที่ 4.2 แสดงตัวดำเนินการสัมพันธ์

ตัวดำเนินการ	จุดประสงค์
<	น้อยกว่า
<=	น้อยกว่าหรือเท่ากับ
>	มากกว่า
>=	มากกว่าหรือเท่ากับ

ตารางที่ 4.3 แสดงตัวดำเนินการเสมอภาค

ตัวดำเนินการ	จุดประสงค์
<code>==</code>	เท่ากับ
<code>!=</code>	ไม่เท่ากับ

ตารางที่ 4.4 แสดงตัวดำเนินการตรรกยะ

ตัวดำเนินการ	จุดประสงค์
<code>&&</code>	and (และ)
<code> </code>	or (หรือ)
<code>!</code>	not (นิเสธ)

ตารางที่ 4.5 แสดงผลการเชื่อมตัวดำเนินการแบบ `&&` (and)

ตัวถูกดำเนินการ 1	ตัวถูกดำเนินการ 2	ผลที่ได้
ไม่ใช่ 0 (true)	ไม่ใช่ 0 (true)	1 (true)
ไม่ใช่ 0 (true)	0 (false)	0 (false)
0 (false)	ไม่ใช่ 0 (true)	0 (false)
0 (false)	0 (false)	0 (false)

ตารางที่ 4.6 แสดงผลการเชื่อมตัวดำเนินการแบบ `||` (or)

ตัวถูกดำเนินการ 1	ตัวถูกดำเนินการ 2	ผลที่ได้
ไม่ใช่ 0 (true)	ไม่ใช่ 0 (true)	1 (true)
ไม่ใช่ 0 (true)	0 (false)	1 (true)
0 (false)	ไม่ใช่ 0 (true)	1 (true)
0 (false)	0 (false)	0 (false)

ตารางที่ 4.7 แสดงผลตัวดำเนินการแบบ `!` (not)

ตัวถูกดำเนินการ	<code>!</code> ตัวถูกดำเนินการ
ไม่ใช่ 0 (true)	0 (false)
0 (false)	1 (true)

ตารางที่ 4.8 แสดงผลตัวดำเนินการที่มีความสมมูลกัน

! (a == b)	a != b
! (a != b)	a == b
! (a < b)	a >= b
! (a <= b)	a > b
! (a > b)	a <= b
! (a >= b)	a < b
! (Expression_1 && Expression_2)	(! Expression_1 ! Expression_2)
! (Expression_1 Expression_2)	(! Expression_1 && ! Expression_2)

4.2 ตัวดำเนินการเอกภาค (unary operators)

จะใช้กับตัวดำเนินการเพียงจำนวนเดียว ซึ่งตัวดำเนินการเอกภาคโดยปกติแล้วจะอยู่หน้าตัวถูกดำเนินการ แต่ก็อาจจะอยู่ด้านหลังก็ได้ การดำเนินการชนิดเอกภาคที่เรารู้จักคือ จะมีเครื่องหมาย ลบ (-) นำหน้าค่าคงตัวแบบจำนวน, ตัวแปร หรือ นิพจน์

ตัวอย่างที่ 4.1 แสดงการดำเนินการชนิดเอกภาค (-)

-743, -0X7F, -0.2, -5E+8, -X, -(X+Y)

ในภาษา C มีตัวดำเนินการเอกภาค อีก 2 ตัว คือ

- 1) ตัวดำเนินการเพิ่มขึ้น (++) และ
- 2) ตัวดำเนินการลดลง (--)

4.3 นิพจน์ (expression)

อาจจะเป็นตัวเลขหรือตัวอักขระ หรืออาจจะมี การนำตัวดำเนินการมากระทำระหว่างตัวเลขหรือตัวอักขระก็ได้

ตัวอย่างที่ 4.2 แสดงถึงนิพจน์ต่าง ๆ

- 1) (a - b)
- 2) x + y
- 3) ++i
- 4) a * (b - c)
- 5) (a - b) / (a + b)
- 6) x + 2

4.4 ข้อความสั่งกำหนดค่า (assignment statement)

จะเป็นการกำหนดค่าให้กับตัวแปรทางด้านซ้ายมือ ซึ่งมีรูปแบบทั่วไป คือ

identifier = expression ;

ซึ่ง identifier โดยทั่วไปเป็นตัวแปร

ส่วน expression อาจจะเป็น ค่าคงตัว, ตัวแปร หรือนิพจน์

ตัวอย่างที่ 4.3 จะเป็นการกำหนดค่าที่ถูกต้อง

$a = 3$; หมายถึง จะนำค่าจำนวนเต็ม 3 ไปไว้ในตัวแปร a

$x = y$; หมายถึง จะนำค่าของตัวแปร y ไปไว้ในตัวแปร x

$sum = a + b$; หมายถึง จะมีการคำนวณค่าของนิพจน์ $a + b$ แล้วนำผลลัพธ์ไปไว้ในตัวแปร sum

ตัวอย่างที่ 4.4 จะเป็นการกำหนดค่าที่ไม่ถูกต้อง

$3 = answer$; เนื่องจากทางด้านซ้ายมือต้องเป็นตัวแปร

$answer + 3 = answer$; เนื่องจากทางด้านซ้ายมือเป็นการตั้งชื่อที่ผิด

ตัวอย่างที่ 4.5 กำหนด $int\ a$;

:

$a = 20$;

จะเป็นการนำค่า 20 ไปไว้ในหน่วยความจำของตัวแปร a โดยถ้าหน่วยความจำของตัวแปร a มีค่าเท่าไร ก็จะถูกแทนค่าด้วย 20 ทำให้ข้อมูลเดิมถูกทำลาย

ตัวอย่างที่ 4.6 กำหนด $float\ KMS_PER_MILE, miles, kms$;

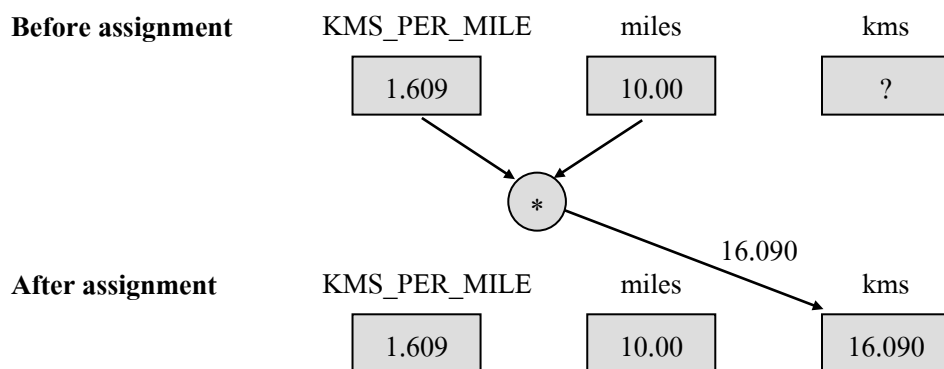
:

$KMS_PER_MILE = 1.609$;

$miles = 10.00$;

$kms = KMS_PER_MILE * miles$;

จะได้ผลของตัวแปรก่อนและหลังกำหนดค่า ดังรูปที่ 4.1



รูปที่ 4.1

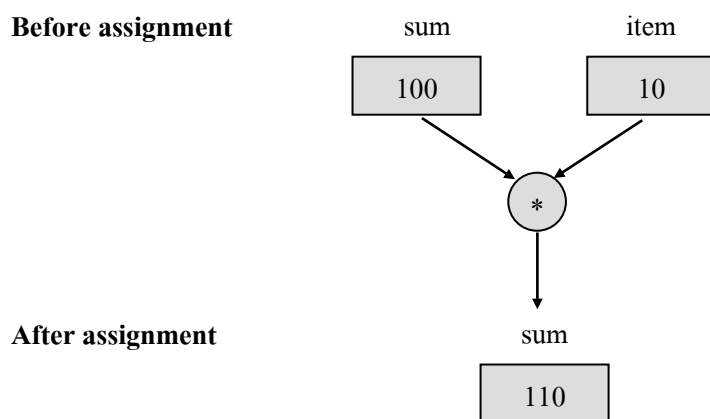
ตัวอย่างที่ 4.7 กำหนด

```

int sum, item ;
:
sum = 100 ;
item = 10 ;
sum = sum + item ;

```

จะได้ผลของตัวแปรก่อนและหลังกำหนดค่า ดังรูปที่ 4.2



รูปที่ 4.2

ในบางครั้งเราอาจจะกำหนดค่าเริ่มต้นให้กับตัวแปรในส่วนที่ประกาศตัวแปรได้

ตัวอย่างที่ 4.8 กำหนด `int C = 12` ; หมายความว่า ตัวแปร C ที่มีข้อมูลชนิดจำนวนเต็ม กำหนดค่าเริ่มต้น = 12

ตัวอย่างที่ 4.9 กำหนด `char star = '*'`; หมายความว่า ตัวแปร `star` ที่มีข้อมูลชนิดอักขระ กำหนดค่าเริ่มต้น = `'*'`

ตัวอย่างที่ 4.10 กำหนด `char text[20] = "Bandid"` ;
หมายความว่า ตัวแปร `text` ที่มีข้อมูลชนิดแถวลำดับของอักขระ กำหนดค่าเริ่มต้นเป็น `"Bandid"`
รายละเอียดของแถวลำดับจะกล่าวถึงในภายหลัง

ตัวอย่างที่ 4.11 กำหนด `int ans ;`

:

`ans = ans + 5 ;`

ในภาษา C สามารถย่อเขียนได้เป็น

`ans += 5 ;`

ในทำนองเดียวกัน

`ans = ans - 5 ;` ย่อเป็น `ans -= 5 ;`

`ans = ans * 5 ;` ย่อเป็น `ans *= 5 ;`

`ans = ans / 5 ;` ย่อเป็น `ans /= 5 ;`

`ans = ans % 5 ;` ย่อเป็น `ans %= 5 ;`

ในภาษา C ก้อนชุดให้มีการกำหนดค่าให้กับตัวแปรหลายๆ ตัวพร้อมกันได้

`identifier1 = identifier2 = ... = expression ;`

ซึ่งในการกำหนดค่านั้น จะเริ่มจากขวามือไปยังซ้ายมือ

ตัวอย่างที่ 4.12 กำหนด `int a, b ;`

`a = b = 5 ;`

จะหมายถึง `b = 5 ;`

`a = b ;`

4.5 ตัวดำเนินการเอกภาคแบบเพิ่มขึ้นและลดลง ครั้งละ 1

การเพิ่มค่าขึ้น 1 หรือลดลง 1 ก่อนหรือหลัง ในภาษา C จะให้ความหมาย ดังนี้

4.5.1 `x++` จะหมายถึง `x` เพิ่มขึ้น 1 หลังจาก `x` ถูกดำเนินการแล้ว

เราจะเรียกว่า `post_incrementing`

ตัวอย่างที่ 4.13 กำหนด $i = 3$;
 $x = i++$;
 หมายถึง $x = i$;
 $i = i + 1$;
 จะได้ผลลัพธ์ ดังนี้ $x = 3, i = 4$

4.5.2 $++x$ หมายถึง เพิ่มค่า x ขึ้น 1 ก่อน x ถูกดำเนินการ
 เราจะเรียกว่า pre_incrementing

ตัวอย่างที่ 4.14 กำหนด $i = 3$;
 $x = ++i$;
 หมายถึง $i = i + 1$;
 $x = i$;
 จะได้ผลลัพธ์ ดังนี้ $x = 4, i = 4$

4.5.3 $x--$ หมายถึง ลดค่า x ลง 1 หลังจาก x ถูกดำเนินการแล้ว
 เราจะเรียกว่า post_decrementing

ตัวอย่างที่ 4.15 กำหนด $i = 3$;
 $x = i--$;
 หมายถึง $x = i$;
 $i = i - 1$;
 จะได้ผลลัพธ์ ดังนี้ $x = 3, i = 2$

4.5.4 $--x$ หมายถึง ลดค่า x ลง 1 ก่อน x ถูกดำเนินการ
 เราจะเรียกว่า pre_decrementing

ตัวอย่างที่ 4.16 กำหนด $i = 3$;
 $x = --i$;
 หมายถึง $i = i - 1$;
 $x = i$;
 จะได้ผลลัพธ์ ดังนี้ $x = 2, i = 2$

ตารางที่ 4.8 แสดงลำดับความสำคัญในการกระทำตัวดำเนินการ

(C OPERATORS AND OPERATOR PRECEDENCE)

ระดับความสำคัญ (PRECEDENCE LEVEL)	ชื่อตัวดำเนินการ (OPERATOR NAME)	สัญลักษณ์ (SYMBOL)	ทิศทางของการดำเนินการ (ASSOCIATIVITY)
1	Function call	()	→
	Array subscripting	[]	→
	Component selection	.	→
	Indirect component selection	->	→
2	Unary plus	+	←
	Unary minus	-	←
	Increment	++	←
	Decrement	--	←
	Logical <i>not</i>	!	←
	Complement	~	←
	Indirection	*	←
	Address of	&	←
	Cast	(types)	←
	Size of type or object	sizeof	←
3	Multiplication	*	→
	Division	/	→
	Modulus (remainder)	%	→
4	Addition	+	→
	Subtraction	-	→
5	Left shift	<<	→
	Right shift	>>	→
6	Less than	<	→
	Less than or equal to	<=	→
	Greater than	>	→
	Greater than or equal to	>=	→
7	Equal to	==	→
	Not equal to	!=	→

ระดับความสำคัญ (PRECEDENCE LEVEL)	ชื่อตัวดำเนินการ (OPERATOR NAME)	สัญลักษณ์ (SYMBOL)	ทิศทางของการดำเนินการ (ASSOCIATIVITY)
8	Bitwise and	&	→
9	Bitwise exclusive or	^	→
10	Bitwise inclusive or		→
11	Logical and	&&	→
12	Logical inclusive or		→
13	Conditional operator	? :	←
14	Simple assignment	=	←
	Assign sum	+=	←
	Assign difference	-=	←
	Assign product	*=	←
	Assign division	/=	←
	Assign remainder	%=	←
	Assign left shift	<=	←
	Assign right shift	>=	←
	Assign and	&=	←
	Assign inclusive or	=	←
	Assign exclusive or	^=	←
15	Comma	,	→

หมายเหตุ 1. ค่าของ precedence level 1 จะมีลำดับความสำคัญสูงสุด

ค่าของ precedence level 15 จะมีลำดับความสำคัญต่ำสุด

2. ใน precedence level เดียวกัน ถ้ามีตัวดำเนินการหลายตัว ถือว่ามีลำดับความสำคัญ

เท่ากัน ทิศทางในการกระทำให้ออกจากเครื่องหมาย → หรือ ← กล่าวคือ

เครื่องหมาย → ทำจากซ้ายมือไปยังขวามือ

เครื่องหมาย ← ทำจากขวามือไปยังซ้ายมือ

ตัวอย่างต่อไปนี้จะคำนึงถึงลำดับความสำคัญของตัวดำเนินการ

ตัวอย่างที่ 4.17 กำหนด $i = 3, c = 10$ จงคำนวณค่า x

จากเงื่อนไขต่อไปนี้

$$4.17.1) \ x = i++;$$

$$4.17.2) \ x = ++i;$$

$$4.17.3) \ x = c++;$$

$$4.17.4) \ x = --c + 2;$$

$$4.17.5) \ x = i-- + ++c;$$

$$4.17.1) \ x = i++;$$

หมายถึง $x = i;$ แทนค่า $i = 3$

$$i = i + 1;$$

ดังนั้น $x = 3$

$$4.17.2) \ x = ++i;$$

หมายถึง $i = i + 1;$ แทนค่า $i = 3$

$$x = i;$$

ดังนั้น $x = 4$

$$4.17.3) \ x = c++;$$

หมายถึง $x = c;$ แทนค่า $c = 10$

$$c = c + 1;$$

ดังนั้น $x = 10$

$$4.17.4) \ x = --c + 2;$$

หมายถึง $c = c - 1;$ แทนค่า $c = 10$

$$x = c + 2; \text{ แทนค่า } c = 9$$

ดังนั้น $x = 9 + 2 = 11$

$$4.17.5) \ x = i-- + ++c;$$

หมายถึง $i = i;$ แทนค่า $i = 3$

$$c = c + 1; \text{ แทนค่า } c = 10$$

$$x = i + c; \text{ แทนค่า } i = 3, c = 11$$

ดังนั้น $x = 3 + 11 = 14$

ตัวอย่างที่ 4.18 แสดงการแปลงนิพจน์พีชคณิตเป็นนิพจน์ภาษา C

นิพจน์พีชคณิต

นิพจน์ในภาษา C

- | | | |
|--|-----------------------|-------------------------------------|
| 1) $b^2 - 4ac$ | \longleftrightarrow | $b * b - 4 * a * c$ |
| 2) $\frac{a+b}{c+d}$ | \longleftrightarrow | $(a + b) / (c + d)$ |
| 3) $\frac{1}{1+x^2}$ | \longleftrightarrow | $1 / (1 + x * x)$ |
| 4) $\frac{y}{2x + \frac{5}{c+d} + 3y}$ | \longleftrightarrow | $y / (2 * x + 5 / (c + d) + 3 * y)$ |
| 5) $\frac{x+y}{5+5(\frac{x+y}{x})}$ | \longleftrightarrow | $(x + y) / (5 + 5 * ((x + y) / x))$ |
| 6) $\frac{1}{c} + \frac{1-a}{1+b}$ | \longleftrightarrow | $(1 / c) + (1 - a) / (1 + b)$ |

ตัวอย่างต่อไปนี้ จะเป็นการคำนวณหาค่าโดยอาศัยลำดับความสำคัญของตัวดำเนินการมาพิจารณา

ตัวอย่างที่ 4.19

$$x = 2 * 3 * 6$$

ขั้นตอนที่ 1 $x = 6 * 6$ (ทำ $2 * 3$)

ขั้นตอนที่ 2 $x = 36$

ตัวอย่างที่ 4.20

$$x = 30 / 5 / 6$$

ขั้นตอนที่ 1 $x = 6 / 6$ (ทำ $30 / 5$)

ขั้นตอนที่ 2 $x = 1$

ตัวอย่างที่ 4.21

$$x = 30\% \ 5 / 2$$

ขั้นตอนที่ 1	$x = 0 / 2$	(ทำ 30% 5)
--------------	-------------	------------

ขั้นตอนที่ 2	$x = 0$	
--------------	---------	--

ตัวอย่างที่ 4.22

$$x = 2 + 5 - 6$$

ขั้นตอนที่ 1	$x = 7 - 6$	(ทำ 2 + 5)
--------------	-------------	------------

ขั้นตอนที่ 2	$x = 1$	
--------------	---------	--

ตัวอย่างที่ 4.23

$$x = 2 + 3 * 4$$

ขั้นตอนที่ 1	$x = 2 + 12$	(ทำ 3 * 4)
--------------	--------------	------------

ขั้นตอนที่ 2	$x = 14$	
--------------	----------	--

ตัวอย่างที่ 4.24

$$x = 2 * 5 * 5 + 3 * 5 + 7$$

ขั้นตอนที่ 1	$x = 10 * 5 + 3 * 5 + 7$	(ทำ 2 * 5)
--------------	--------------------------	------------

ขั้นตอนที่ 2	$x = 50 + 3 * 5 + 7$	(ทำ 10 * 5)
--------------	----------------------	-------------

ขั้นตอนที่ 3	$x = 50 + 15 + 7$	(ทำ 3 * 5)
--------------	-------------------	------------

ขั้นตอนที่ 4	$x = 65 + 7$	(ทำ 50 + 15)
--------------	--------------	--------------

ขั้นตอนที่ 5	$x = 72$	
--------------	----------	--

ถ้ามีวงเล็บให้ทำงานเล็บก่อน

ตัวอย่างที่ 4.25

$$x = (2 + 5) * 6$$

ขั้นตอนที่ 1	$x = 7 * 6$	(ทำ 2 + 5)
--------------	-------------	------------

ขั้นตอนที่ 2	$x = 42$	
--------------	----------	--

ถ้ามีวงเล็บซ้อนกัน ให้ทำที่วงเล็บในสุดก่อน

ตัวอย่างที่ 4.26

$$x = ((2 + 5) + 7) * 6 * 5$$

ขั้นตอนที่ 1	$x = (7 + 7) * 6 * 5$	(ทำ 2 + 5)
--------------	-----------------------	------------

ขั้นตอนที่ 2	$x = 14 * 6 * 5$	(ทำ 7 + 7)
--------------	------------------	------------

ขั้นตอนที่ 3	$x = 84 * 5$	(ทำ 14 * 6)
--------------	--------------	-------------

ขั้นตอนที่ 4	$x = 420$	
--------------	-----------	--

ตัวอย่างที่ 4.27 กำหนดค่าของ $a=2, b=9, c=5$ และ นิพจน์ $b * b - 4 * a * c \geq 0$ จง

แสดงลำดับขั้นตอนในการคำนวณ และหาผลลัพธ์ด้วย

จากตารางที่ 4.8 ที่แสดงลำดับความสำคัญของตัวดำเนินการจะได้ลำดับในการทำงานตามหมายเลขดังนี้

$$b * b - 4 * a * c \geq 0$$

① ④ ② ③ ⑤

เนื่องจาก $a=2, b=9, c=5$ มีขั้นตอนในการคำนวณมีดังนี้

ขั้นตอนที่ 1	$81 - 4 * a * c \geq 0$	คำนวณหา $b * b$
ขั้นตอนที่ 2	$81 - 8 * c \geq 0$	คำนวณหา $4 * a$
ขั้นตอนที่ 3	$81 - 40 \geq 0$	คำนวณหา $8 * c$
ขั้นตอนที่ 4	$41 \geq 0$	คำนวณหา $81 - 40$
ขั้นตอนที่ 5	1	เนื่องจาก $41 \geq 0$ เป็นจริง

ตัวอย่างที่ 4.28 กำหนดค่าของ $a=10, b=3, c=7$ และ นิพจน์ $a + b \geq 3 * c == a ! = 2 * c + b$

จงแสดงลำดับขั้นตอนในการคำนวณ และหาผลลัพธ์ด้วย

จากตารางที่ 4.8 จะได้ลำดับในการทำงานตามหมายเลขดังนี้

$$a + b \geq 3 * c == a ! = 2 * c + b$$

③ ⑤ ① ⑥ ⑦ ② ④

เนื่องจาก $a=10, b=3, c=7$ มีขั้นตอนในการคำนวณ มีดังนี้

ขั้นตอนที่ 1	$a + b \geq 21 == a ! = 2 * c + b$	คำนวณ $3 * c$
ขั้นตอนที่ 2	$a + b \geq 21 == a ! = 14 + b$	คำนวณ $2 * c$
ขั้นตอนที่ 3	$13 \geq 21 == a ! = 14 + b$	คำนวณ $a + b$
ขั้นตอนที่ 4	$13 \geq 21 == a ! = 17$	คำนวณ $14 + b$
ขั้นตอนที่ 5	$0 == a ! = 17$	คำนวณ $13 \geq 21$
ขั้นตอนที่ 6	$0 ! = 17$	คำนวณ $0 == a$
ขั้นตอนที่ 7	1	คำนวณ $0 ! = 17$

ตัวอย่างที่ 4.29 กำหนดค่าของ $a = 10, b = 3, c = 7$ และ นิพจน์ $(a + b >= 3 * c) == (a != 2 * c + b)$

จงแสดงลำดับขั้นตอนในการคำนวณ และหาผลลัพธ์ด้วย

จากตารางที่ 4.8 จะได้ลำดับในการทำงานตามหมายเลขดังนี้

$$(a + b >= 3 * c) == (a != 2 * c + b)$$

② ③ ① ⑦ ⑥ ④ ⑤

เนื่องจาก $a = 10, b = 3, c = 7$ มีขั้นตอนในการคำนวณ มีดังนี้

ขั้นตอนที่ 1	$(a + b >= 21) == (a != 2 * c + b)$	คำนวณ $3 * c$
ขั้นตอนที่ 2	$(13 >= 21) == (a != 2 * c + b)$	คำนวณ $a + b$
ขั้นตอนที่ 3	$0 == (a != 2 * c + b)$	คำนวณ $13 >= 21$
ขั้นตอนที่ 4	$0 == (a != 6 + b)$	คำนวณ $2 * c$
ขั้นตอนที่ 5	$0 == (10 != 13)$	คำนวณ $6 + b$
ขั้นตอนที่ 6	$0 = 1$	คำนวณ $10 != 13$
ขั้นตอนที่ 7	0	คำนวณ $0 = 1$

ตัวอย่างที่ 4.30 กำหนดค่าของ $a = 10, b = 20, c = 15, d = 8$ และ $e = 40$

และ นิพจน์ $(a + b / (c - 5)) / ((d + 7) / (e - 37) \% 3)$

จงแสดงลำดับขั้นตอนในการคำนวณ และหาผลลัพธ์ด้วย

จากตารางที่ 4.8 จะได้ลำดับในการทำงานตามหมายเลขดังนี้

$$(a + b / (c - 5)) / ((d + 7) / (e - 37) \% 3)$$

⑤ ④ ① ⑧ ② ⑥ ③ ⑦

เนื่องจาก $a = 10, b = 20, c = 15, d = 8$ และ $e = 40$ มีขั้นตอนในการคำนวณ ดังนี้

ขั้นตอนที่ 1	$(a + b / 10) ((d + 7) / (e - 37) \% 3)$	คำนวณ $(c - 5)$ ได้ 10
ขั้นตอนที่ 2	$(a + b / 10) (15 / (e - 37) \% 3)$	คำนวณ $(d + 7)$ ได้ 15
ขั้นตอนที่ 3	$(a + b / 10) (15 / 3 \% 3)$	คำนวณ $(e - 37)$ ได้ 3
ขั้นตอนที่ 4	$(a + 2) / (15 / 3 \% 3)$	คำนวณ $b / 10$ ได้ 2
ขั้นตอนที่ 5	$12 / (15 / 3 \% 3)$	คำนวณ $(a + 2)$ ได้ 12
ขั้นตอนที่ 6	$12 / (5 \% 3)$	คำนวณ $(15 / 3)$ ได้ 5
ขั้นตอนที่ 7	$12 / 2$	คำนวณ $(5 \% 3)$ ได้ 2
ขั้นตอนที่ 8	6	คำนวณ $(12 / 2)$ ได้ 6

และคำนวณหาผลลัพธ์ได้คือ 6

แต่ถ้าตัวอย่างนี้ไม่มีวงเล็บเลย จะมีลำดับในการทำงานตามหมายเลข ดังนี้

$$a + b / c - 5 / d + 7 / e - 37 \% 3$$

⑤ ① ⑥ ② ⑦ ③ ⑧ ④

และคำนวณหาผลลัพธ์ได้คือ 10

การนำข้อมูลชนิดจำนวนเต็ม มากระทำด้วยตัวดำเนินการคำนวณ คือ $+$, $-$, $*$, $/$ และ $\%$ จะให้ผลลัพธ์เป็นจำนวนเต็ม

ตัวอย่างที่ 4.31 กำหนด $a = 10, b = 3$

- 1) $a + b = 10 + 3 = 13$
- 2) $a - b = 10 - 3 = 7$
- 3) $a * b = 10 * 3 = 30$
- 4) $a / b = 10 / 3 = 3$ (กรณีนี้จะมีการตัดเศษ)
- 5) $a \% b = 10 \% 3 = 1$

ตัวอย่างที่ 4.32 กำหนด $a = 11, b = -3$

- 1) $a + b = 11 + (-3) = 8$
- 2) $a - b = 11 - (-3) = 11 + 3 = 14$
- 3) $a * b = 11 * (-3) = -33$
- 4) $a / b = 11 / (-3) = -3$ (กรณีนี้จะมีการตัดเศษ)
- 5) $a \% b = 11 \% (-3) = 2$

หมายเหตุ 1) การคูณและการหารจำนวนเต็ม ยังคงใช้หลักทางพีชคณิต กล่าวคือ

จำนวนเต็มบวกคูณจำนวนเต็มบวก ผลลัพธ์จะเป็นจำนวนเต็มบวก

จำนวนเต็มลบคูณจำนวนเต็มลบ ผลลัพธ์จะเป็นจำนวนเต็มบวก

จำนวนเต็มบวกคูณจำนวนเต็มลบ ผลลัพธ์จะเป็นจำนวนเต็มลบ

จำนวนเต็มบวกหารด้วยจำนวนเต็มบวก ผลลัพธ์จะเป็นจำนวนเต็มบวก

จำนวนเต็มลบหารด้วยจำนวนเต็มลบ ผลลัพธ์จะเป็นจำนวนเต็มบวก

จำนวนเต็มบวกหารด้วยจำนวนเต็มลบ ผลลัพธ์จะเป็นจำนวนเต็มลบ

จำนวนเต็มลบหารด้วยจำนวนเต็มบวก ผลลัพธ์จะเป็นจำนวนเต็มลบ

และ สำหรับการหารด้วยจำนวนเต็มศูนย์ จะหาค่าไม่ได้

เช่น $11/0$ หาค่าไม่ได้

2) การหาเศษที่ได้จากการหารระหว่างจำนวนเต็มบวกและจำนวนเต็มลบ ผลลัพธ์ของเครื่องหมายของเศษที่ได้ ในภาษา C ยังไม่ค่อยชัดเจน แต่ในภาษา C บางรุ่น จะกำหนดหลักเกณฑ์ ดังนี้

(1) ถ้า a เป็นจำนวนเต็มบวก และ b เป็นจำนวนเต็มบวก ผลลัพธ์ของเครื่องหมายของเศษที่ได้จะเหมือนกับตัวตั้ง (a) เช่น $11 \% 3 = 2$, $a = 11$, $b = 3$

(2) ถ้า a เป็นจำนวนเต็มบวก และ b เป็นจำนวนเต็มลบ ผลลัพธ์ของเครื่องหมายของเศษที่ได้จะเหมือนกับตัวตั้ง (a) เช่น $11 \% (-3) = 2$, $a = 11$, $b = -3$

(3) ถ้า a เป็นจำนวนเต็มลบ และ b เป็นจำนวนเต็มบวก ผลลัพธ์ของเครื่องหมายของเศษที่ได้จะเหมือนกับตัวตั้ง (a) เช่น $(-11) \% 3 = -2$, $a = -11$, $b = 3$

(4) ถ้า a เป็นจำนวนเต็มลบ และ b เป็นจำนวนเต็มลบ ผลลัพธ์ของเครื่องหมายของเศษที่ได้จะเหมือนกับตัวตั้ง (a) เช่น $(-11) \% (-3) = -2$

สรุป การหาเศษผลลัพธ์ของเครื่องหมายของเศษที่ได้จะเหมือนกับตัวตั้งในทุกกรณี และถ้าตัวหารเป็นจำนวนเต็มศูนย์ จะหาค่าไม่ได้ เช่น $11 \% 0$ หาค่าไม่ได้

การนำข้อมูลชนิดจำนวนจุดลอยตัวมากระทำด้วยตัวดำเนินการคำนวณ คือ $+$, $-$, $*$, $/$ และ $\%$ ผลลัพธ์ที่ได้จะเป็นจำนวนจุดลอยตัว แต่ในการหาเศษเราจะอาศัยฟังก์ชัน `fmod` ซึ่งเป็นไลบรารีฟังก์ชัน เวลานำฟังก์ชันนี้มาใช้เราต้องกำหนด `#include <math.h>` ไว้ที่ตอนต้นของโปรแกรม เช่น `fmod(13.657, 2.333) = 1.992` เครื่องหมายของเศษจะเหมือนกับเครื่องหมายของตัวตั้ง

ตัวอย่างที่ 4.33 กำหนด $a = 5.0$, $b = 2.0$

$$1) a + b = 5.0 + 2.0 = 7.0$$

$$2) a - b = 5.0 - 2.0 = 3.0$$

$$3) a * b = 5.0 * 2.0 = 10.0$$

$$4) a / b = 5.0 / 2.0 = 2.5$$

ความสัมพันธ์ระหว่างตัวดำเนินการแบบ $/$ และ $\%$

ตัวอย่างที่ 4.34 แสดงความสัมพันธ์ระหว่างตัวดำเนินการแบบ / และ % โดยพิจารณาจาก
หาร 7 ด้วย 2 แบบ หารยาว ดังนี้

$$\begin{array}{r} 7 / 2 \\ \downarrow \\ 2 \overline{) 7} \\ \underline{6} \\ 1 \end{array} \leftarrow 7 \% 2$$

โดยที่ 7 เป็นตัวตั้ง, 2 เป็นตัวหาร, 3 เป็นผลหาร และ 1 เป็นเศษ นำมาเขียนความสัมพันธ์ได้ดังนี้

$$\text{ตัวตั้ง} = (\text{ผลหาร}) (\text{ตัวหาร}) + \text{เศษ}$$

$$\begin{aligned} \text{และ} \quad 7 &= (7 / 2) * 2 + (7 \% 2) \\ &= 3 * 2 + 1 \end{aligned}$$

ถ้าให้ m เป็นตัวตั้ง และ n เป็นตัวหาร นำมาเขียนเป็นความสัมพันธ์ทั่วไปได้ดังนี้

$$m = (m / n) * n + (m \% n)$$

ในกรณีนำข้อมูลต่างชนิดกันมากระทำด้วยตัวดำเนินการคำนวณ เช่น +, -, *, / และ % ก่อนที่จะ
ได้ผลลัพธ์สุดท้าย คอมไพเลอร์จะมีการแปลงชนิดของข้อมูลที่น่ามากระทำต่อกันให้เป็นประเภท
เดียวกันเสียก่อน เช่น

$$2 + 3.0 \text{ ต้องแปลงเป็น } 2.0 + 3.0 \text{ (แปลง 2 เป็น 2.0) ก่อน}$$

ในภาษา C ได้กำหนดตำแหน่ง (rank) ของชนิดข้อมูลจากน้อยไปมาก ดังนี้

$$\text{char} < \text{int} < \text{long} < \text{float} < \text{double}$$

และกำหนดกฎเกณฑ์ในการแปลงดังนี้

- 1) ถ้าตัวถูกดำเนินการตัวหนึ่งเป็นข้อมูลชนิด long double และอีกตัวหนึ่งจะต้องถูกแปลงให้เป็น
ข้อมูลชนิด long double และผลลัพธ์ที่ได้จะเป็นข้อมูลชนิด long double
- 2) ถ้าไม่ใช่ข้อมูลในแบบข้อ 1) ถ้าตัวถูกดำเนินการตัวหนึ่งเป็นข้อมูลชนิด double และอีกตัวหนึ่ง
จะต้องถูกแปลงเป็นข้อมูลชนิด double และอีกตัวหนึ่งจะต้องถูกแปลงเป็นข้อมูลชนิด double
และผลลัพธ์ที่ได้จะเป็นข้อมูลชนิด double

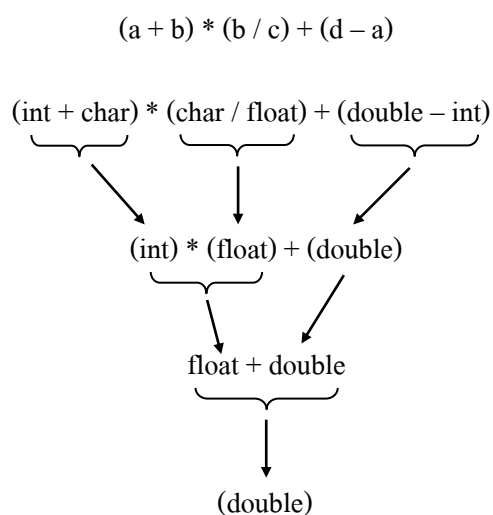
- 3) ถ้าไม่ใช่ข้อมูลในแบบ ข้อ 2) ถ้าตัวถูกดำเนินการตัวหนึ่งเป็นข้อมูลชนิด float และอีกตัวหนึ่งจะต้องถูกแปลงเป็นข้อมูลชนิด float และผลลัพธ์ที่ได้จะเป็นข้อมูลชนิด float
- 4) ถ้าไม่ใช่ข้อมูลในแบบ ข้อ 3) ถ้าตัวถูกดำเนินการตัวหนึ่งเป็นข้อมูลชนิด unsigned long int และอีกตัวหนึ่งจะต้องถูกแปลงเป็นข้อมูลชนิด unsigned long int และผลลัพธ์จะเป็น unsigned long int
- 5) ถ้าไม่ใช่ข้อมูลในแบบ ข้อ 4) ถ้าตัวถูกดำเนินการตัวหนึ่งเป็นข้อมูลชนิด long int และตัวถูกดำเนินการอีกตัวหนึ่งเป็นข้อมูลชนิด unsigned int แล้ว
 - 5.1) ถ้า unsigned int สามารถแปลงเป็น long int ได้ ตัวถูกดำเนินการชนิด unsigned int จะถูกแปลงและผลลัพธ์จะเป็นข้อมูลชนิด long int
 - 5.2) มิฉะนั้นแล้ว ตัวถูกดำเนินการทั้งสองนี้ จะถูกแปลงเป็น unsigned long int และผลลัพธ์จะเป็นข้อมูลแบบ unsigned long int
- 6) ถ้าไม่ใช่ข้อมูลในแบบ ข้อ 5) ถ้าตัวถูกดำเนินการตัวหนึ่งเป็นข้อมูลชนิด long int และตัวถูกดำเนินการอีกตัวหนึ่งจะถูกแปลงเป็น long int และผลลัพธ์จะเป็น long int
- 7) ถ้าไม่ใช่ข้อมูลในแบบ ข้อ 6) ตัวถูกดำเนินการตัวหนึ่งเป็นข้อมูลชนิด unsigned int และตัวถูกดำเนินการอีกตัวหนึ่งจะถูกแปลงเป็น unsigned int และผลลัพธ์จะเป็น unsigned int
- 8) ถ้าไม่ใช่เงื่อนไขด้านบนทั้งหมด ตัวถูกดำเนินการจะถูกแปลงเป็นชนิด int และผลลัพธ์เป็น int

ตัวอย่างที่ 4.35 จะเป็นการแสดงการแปลงข้อมูลให้สอดคล้องกับกฎเกณฑ์การแปลง

กำหนด `int a ;`
 `char b ;`
 `float c ;`
 `double d ;`

และนิพจน์ $(a + b) * (b / c) + (d - a)$

สามารถแสดงการแปลงได้ดังนี้



ตัวอย่างที่ 4.36 ในการคำนวณค่าของนิพจน์ที่มีข้อมูลต่างชนิด จะต้องมีการแปลงชนิดข้อมูลให้ถูกต้องก่อน แล้วนำไปเก็บไว้ในตัวแปร

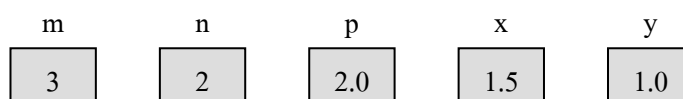
กำหนด `int m, n ;`
 `double p, x, y ;`
 `!`
 `m = 3 ;`
 `n = 2 ;`
 `p = 2.0 ;`
 `x = m / p ;`
 `y = m / n ;`

จาก $x = m / p$;

m มีค่า 3 จะถูกแปลงเป็น 3.0

ดังนั้น $m / p = 3.0 / 2.0 = 1.5$ นำค่า 1.5 ไปไว้ในตัวแปร x ที่เป็นชนิด double

ส่วน $y = m / n$ จะได้ว่า $3 / 2 = 1$ ก่อนจะนำไปไว้ในตัวแปร y ที่เป็นชนิด double ต้องมีการแปลง 1 เป็น 1.0 ก่อน แล้วถึงไปเก็บไว้ในตัว y ดังรูปที่ 4.3



รูปที่ 4.3

ตัวอย่างที่ 4.37 ถ้ากำหนดชนิดของข้อมูลในการเก็บผลลัพธ์ไม่สอดคล้องกัน จะทำให้ผลลัพธ์ที่ได้จะมีการตัดเศษออก เช่น

```
กำหนด      int y ;

            double x ;

            x = 9 * 0.5 ;

            y = 9 * 0.5 ;
```

เนื่องจาก $9 * 0.5 = 4.5$ และ x มีข้อมูลชนิด double ดังนั้น จะนำ 4.5 ไปเก็บไว้ในตัวแปร x ได้ แต่ y มีข้อมูลชนิดจำนวนเต็ม ดังนั้น จะนำ 4 ไปเก็บไว้ในตัวแปร y ส่วนทศนิยมจะถูกตัดทิ้งไป ดังรูปที่ 4.4



รูปที่ 4.4

ค่าของข้อมูลในนิพจน์ เราสามารถเปลี่ยนได้ตามที่กำหนดได้เอง โดยการนำหน้าด้วยชื่อของชนิดข้อมูลที่ต้องการจะแปลงโดยปิดล้อมด้วยวงเล็บ ซึ่งจะมีความหมายทั่วไปดังนี้

(data types) expression

โครงสร้างแบบนี้ เราจะเรียกว่า cast หรือ type cast

ตัวอย่างที่ 4.38 กำหนด `int i = 7 ;`

`float f = 8.5 ;`

นิพจน์ `(i + f) % 4` ไม่ถูกต้อง เนื่องจากนิพจน์ `(i + f)` เป็นข้อมูลชนิดลอยตัว แทนที่จะเป็นจำนวนเต็ม ถ้าต้องการแปลงให้เป็นจำนวนเต็ม ให้เติม `(int)` หน้านิพจน์ ดังนี้

`((int)(i + f)) % 4`

และผลลัพธ์ที่ได้ จะมีเศษ = 3

ตัวอย่างที่ 4.39 กำหนด `double first = 4.7 ;`

`int second = 27 ;`

ลำดับที่	ข้อความสั่ง	ค่าหลังจากกระทำการ (exection) ของ	
		นิพจน์	ตัวแปร first
1	<code>(int) (first + second) ;</code>	31	4.7
2	<code>first = (int) first + second ;</code>	31.0	31.0
3	<code>first = (int) first % second ;</code>	4.0	4.0
4	<code>first = second % (int) first ;</code>	3.0	3.0

ตัวอย่างที่ 4.40 กำหนด `double first, second ;` จงคำนวณหาค่าของตัวแปร second เมื่อ กำหนดค่าเริ่มต้นของ `first = 12.75` และ `second = 13.75`

ลำดับที่	ข้อความสั่ง	ค่าของ second หลังกระทำการ
1	<code>second += first ;</code>	26.5
2	<code>second -= first ;</code>	1.0
3	<code>second *= first ;</code>	175.3125
4	<code>second /= first ;</code>	1.078431

ตัวอย่างที่ 4.41 กำหนด `int third, fourth;` จงคำนวณหาค่าของตัวแปร `fourth` เมื่อกำหนดค่าเริ่มต้นของ `third = 13` และ `fourth = 20`

ลำดับที่	ข้อความสั่ง	ค่าของ <code>fourth</code> หลังกระทำการ
1	<code>fourth += third;</code>	33
2	<code>fourth -= third;</code>	7
3	<code>fourth *= third;</code>	260
4	<code>fourth /= third;</code>	1
5	<code>fourth %= third;</code>	7
6	<code>fourth += third + 4;</code>	37
7	<code>fourth -= third + 4;</code>	3
8	<code>fourth *= third + 4;</code>	340
9	<code>fourth /= third + 4;</code>	1
10	<code>fourth %= third + 4;</code>	3

ตัวอย่างที่ 4.42 กำหนดนิพจน์ดังต่อไปนี้ โดย `x = 20` และ `y = 4`

$$x + y >= 13 \ \&\& \ ! \ (x - y) \ || \ x * y - 16 == 4$$

จงแสดงลำดับขั้นตอนในการคำนวณและหาผลลัพธ์ด้วย

จากตารางที่ 4.8 จะได้ลำดับในการทำงานตามหมายเลขดังนี้

$$x + y >= 13 \ \&\& \ ! \ (x - y) \ || \ x * y - 16 == 4$$

④ ⑥ ⑧ ② ① ⑨ ③ ⑤ ⑦

ขั้นตอนที่	ตัวดำเนินการ	นิพจน์
1	–	<code>x + y >= 13 && ! 16 x * y - 16 == 4</code>
2	!	<code>x + y >= 13 && 0 x * y - 16 == 4</code>
3	*	<code>x + y >= 13 && 0 80 - 16 == 4</code>
4	+	<code>24 >= 13 && 0 80 - 16 == 4</code>
5	–	<code>24 >= 13 && 0 64 == 4</code>
6	>=	<code>1 && 0 64 == 4</code>
7	==	<code>1 && 0 0</code>
8	&&	<code>0 0</code>
9		<code>0</code>

ตัวอย่างที่ 4.43 กำหนด `int = var1 = 15, var2 = 5, is_done = 1 ;`
`double var3 = 3.5 ;`
`char var4 = 'c' ;`

จงพิจารณาผลลัพธ์ที่ได้ตารางด้านล่างนี้ โดยใช้ตารางที่ 4.8

ลำดับที่	นิพจน์	ผลลัพธ์
1	<code>(var1 <= 10) && (var2 == 5)</code>	0
2	<code>(var1 <= 10) (var2 == 5)</code>	1
3	<code>! var1 ! (! var2)</code>	1
4	<code>! var1 var2</code>	1
5	<code>var1 && var2 && var3</code>	1
6	<code>var1 && (var2 - 5) && var3</code>	0
7	<code>(var3 == 3.5) (var4 == 'c')</code>	1
8	<code>! is_done</code>	0
9	<code>is_done (var4 != 'c')</code>	1
10	<code>! var3 && ! var2</code>	0
11	<code>! (var3 var2)</code>	0

ตัวอย่างที่ 4.44 กำหนดนิพจน์ `(time >= 9) && (time <= 11)`

ถ้ามีเครื่องหมายนิเสธนำหน้า เช่น `! ((time > 9) && (time <= 11))`

จะสมมูลกับ `(! (time >= 9)) || (! (time <= 11))`

จะสมมูลกับ `(time > 9) || (time > 11)`

ตัวอย่างที่ 4.45 นิพจน์ `! ((student_status != 'u') || (grade_point_average < 3.50))`

จะสมมูลกับ `(student_status == 'u') && (grade_point_average >= 3.50)`

ตัวอย่างที่ 4.46 นิพจน์ $! ((a < c) \&\& (b == d))$

จะสมมูลกับ $(a >= c) || (b != d)$

ตัวอย่างที่ 4.47 นิพจน์ $! ((a >= 5) || (b == 7))$

จะสมมูลกับ $(a < 5) \&\& (b != 7)$

ตัวอย่างโปรแกรมที่ 4.1 เป็นโปรแกรมแสดงค่าของตัวแปร ที่มีการกำหนดค่าในส่วนประกาศของโปรแกรม

```
#include <stdio.h>
main ()
{
    int    integer_var = 100 ;
    float  floating_var = 331.79 ;
    double double_var = 8.44e+11 ;
    char   char_var = 'w' ;

    printf ( "integer_var = %d\n", integer_var ) ;
    printf ( "floating_var = %f\n", floating_var ) ;
    printf ( "double_var = %e\n", double_var ) ;
    printf ( "char_var = %c\n", char_var ) ;
}
```

```
integer_var = 100
floating_var = 331.789978
double_var = 8.440000E+11
char_var = W
```


ตัวอย่างโปรแกรมที่ 4.2 เป็นโปรแกรมแสดงการใช้ตัวดำเนินการทางคณิตศาสตร์

```

/* Illustrate the use of various arithmetic operators */
#include <stdio.h>
main ()
{
    int  a = 100 ;
    int  b = 2 ;
    int  c = 25 ;
    int  d = 4 ;
    int  result ;

    result = a - b ;          /* subtraction */
    printf ( "a - b = %d\n", result ) ;

    result = b * c ;          /* multiplication */
    printf ( "b * c = %d\n", result ) ;

    result = a / c ;          /* division */
    printf ( "a / c = %d\n", result ) ;

    result = a + b * c ;      /* precedence */
    printf ( "a + b * c = %d\n", result ) ;

    printf ( "a * b + c * d = %d\n", a * b + c * d ) ;
}

```

```

a - b = 98
b * c = 50
a / c = 4
a + b * c = 150
a * b + c * d = 300

```

ตัวอย่างโปรแกรมที่ 4.3 เป็นโปรแกรมแสดงการคำนวณหาค่านิพจน์เลขคณิต

```
/* More arithmetic expression */
#include <stdio.h>
main ()
{
    int    a = 25 ;
    int    b = 2 ;
    int    result ;
    float  c = 25.0 ;
    float  d = 2.0 ;

    printf ( "6 + a / 5 * b = %d\n", 6 + a / 5 * b );
    printf ( "a / b * b = %d\n", a / b * b );
    printf ( "c / d * d = %f\n", c / d * d );
    printf ( "-a = %d\n", -a );
}
```

```
6 + a / 5 * b = 16
a / b * b = 24
c / d * d = 25.000000
-a = -25
```

ตัวอย่างโปรแกรมที่ 4.4 เป็นโปรแกรมแสดงการใช้ตัวดำเนินการแบบ %

```
/* The modulus operator */
#include <stdio.h>
main ()
{
    int    a = 25, b = 5, c = 10, d = 7 ;

    printf ( "a %% b = %d\n", a % b );
    printf ( "a %% c = %d\n", a % c );
    printf ( "a %% d = %d\n", a % d );
    printf ( "a / d * d + a %% d = %d\n", a / d * d + a % d );
}
```

```
a % b = 0
a % c = 5
a % d = 4
a / d * d + a % d = 25
```

ตัวอย่างโปรแกรมที่ 4.5 เป็นโปรแกรมแสดงการแปลงข้อมูล

```

/* Basic conversions in C */
#include <stdio.h>
main ()
{
    float  f1 = 123.125, f2 ;
    int    i1, i2 = -150 ;
    char   c = 'a' ;

    i1 = f1 ;          /* floating to integer conversion */
    printf ( "%f assigned to an int produces %d\n", f1, i1 ) ;

    f1 = i2 ;          /* interger to floating conversion */
    printf ( "%d assigned to a float produces %f\n", i2, f1 ) ;

    f1 = i2 / 100 ;     /* interger divided by interger */
    printf ( "%d divided by 100 produces %f\n", i2, f1 ) ;

    f2 = i2 / 100.0 ;   /* interger divided by a float */
    printf ( "%d divided by 100.0 produces %f\n", i2, f2 ) ;
}

```

<p>123.125000 assigned to an int produces 123 -150 assigned to a float produces -150.000000 -150 divided by 100 produces -1.000000 -150 divided by 100.00 produces -1.500000</p>

ตัวอย่างโปรแกรมที่ 4.6 เป็นโปรแกรมแสดงการใช้ตัวดำเนินการเพิ่มขึ้นและลดลง

```

/* Preincrementing and postincrementing */
#include <stdio.h>
int main ()
{
    int c ;                /* define variable */

    /* demonstrate postincrement */
    c = 5 ;                /* assign 5 to c */
    printf ( "%d\n", c ) ; /* print 5 */
    printf ( "%d\n", c++ ) ; /* print 5 then postincrement */
    printf ( "%d\n\n", c ) ; /* print 6 */

    /* demonstrate preincrement */
    c = 5 ;                /* assign 5 to c */
    printf ( "%d\n", c ) ; /* print 5 */
    printf ( "%d\n", ++c ) ; /* preincrement then print 6 */
    printf ( "%d\n", c ) ; /* print 6 */

    return 0 ; /* indicate program ended successfully */

} /* end function main */

```

```

5
5
6

5
6
6

```

ตัวอย่างโปรแกรมที่ 4.7 เป็นการคำนวณหารากที่สองโดยอาศัยฟังก์ชัน sqrt

```

/*
 * Performs three square root computations
 */

#include <stdio.h> /* definitions of printf, scanf */
#include <math.h> /* definitions of sqrt */

int main (void)
{
    double first, second, /* input – two data values */
    first_sqrt, /* output – square root of first */
    second_sqrt, /* output – square root of second */
    sum_sqrt, /* output – square root of sum */

    /* Get first number and display its square root. */
    printf ( "Enter the first number> " );
    scanf ( "%lf", &first );
    first_sqrt = sqrt ( first );
    printf ( "The square root of the first number is %.2f\n", first_sqrt );

    /* Get second number and display its square root. */
    printf ( "Enter the second number> " );
    scanf ( "%lf", &second );
    second_sqrt = sqrt ( second );
    printf ( "The square root of the second number is %.2f\n", second_sqrt );

    /* Display the square root of the sum of the two numbers. */
    sum_sqrt = sqrt ( first + second );
    printf ( "The square root of the sum of the two numbers is %.2f\n", sum_sqrt );

    return ( 0 );
}

```

```

Enter the first number > 9.0
The square root of the first number is 3.00
Enter the second number > 16.0
The square root of the second number is 4.00
The square root of the sum of the two numbers is 5.00

```

ตัวอย่างโปรแกรมที่ 4.8 เป็นโปรแกรมนำค่าจำนวนเต็ม 2 จำนวน มาบวกกัน

```

/* This program adds two integer values and displays
   the results */
#include <stdio.h>
int main ()
{
    /* Declare variables */
    int value1, value2, sum ;

    /* Assign vales and compute the result */
    value1 = 50 ;
    value2 = 25 ;
    sum = value1 + value2 ;

    /* Display the result */
    printf ( "The sum of %d and %d is %d\n", value1, value2, sum ) ;
}

```

The sum of 50 and 25 is 75

ตัวอย่างโปรแกรมที่ 4.9 เป็นโปรแกรมการเปลี่ยนหน่วยระยะทางจากไมล์ไปเป็นกิโลเมตรโดยการป้อนข้อมูลเข้าไป

```

/* Converts distances from miles to kilometers. */
#include <stdio.h> /* printf, scanf definitions */
#define KMS_PER_MILE 1.609 /* conversion constant */

int main (void)
{
    double miles, /* distance in miles */
           kms /* equivalent distance in kilometers */

    /* Get and echo the distance in miles. */
    scanf ( "%lf", &miles ) ;
    printf ( "The distance in miles is % .2f. \n", miles ) ;

    /* Convert the distance to kilometers. */
    kms = KMS_PER_MILE * miles ;

    /* Display the distance in kilometers. */
    printf ( "That equals % .2f kilometers. \n", kms ) ;

    return ( 0 ) ;
}

```

The distance in miles is 112.00.
That equals 180.21 kilometers.

ตัวอย่างโปรแกรมที่ 4.10 เป็นโปรแกรมที่รับข้อมูลเป็นรัศมีเข้าไป แล้วคำนวณหาความยาวรอบวงและพื้นที่วงกลม

```

/*
 * Calculates and displays the area and circumference of a circle
 */

#include <stdio.h>
#define PI 3.14159

int main (void)
{
    double radius ; /* input – radius of a circle */
    double area ; /* output – area of a circle */
    double cicum ; /* output – circumference */

    /* Get the circle radius */
    printf ( "Enter radius>" );
    scanf ( "%lf", &radius );

    /* Calculate the area */
    area = PI * radius * radius ;

    /* Calculate the circumference */
    circum = 2 * PI * radius ;

    /* Display the area and circumference */
    printf ( "The area is %.4f\n", area ) ;
    printf ( "The circumference is %.4f\n", circum ) ;

    return ( 0 ) ;
}

```

```

Enter radius> 5.0
The area is 78.5397
The circumference is 31.4159

```

ตัวอย่างโปรแกรมที่ 4.11 เป็นโปรแกรมแสดงการใช้ตัวดำเนินการสัมพันธ์

```
#include <stdio.h>
main ()
{
    float logic_value ;    /* Numeric value of relational expression */

    printf ( "Logic values of the following relations : \n\n" ) ;

    logic_value = (3 > 5) ;
    printf ( " (3 > 5) is %f\n", logic_value ) ;

    logic_value = (5 > 3) ;
    printf ( " (5 > 3) is %f\n", logic_value ) ;

    logic_value = (3 >= 5) ;
    printf ( " (3 >= 5) is %f\n", logic_value ) ;

    logic_value = (15 >= 3 * 5) ;
    printf ( " (15 >= 3 * 5) is %f\n", logic_value ) ;

    logic_value = (8 < (10 - 2) ) ;
    printf ( " (8 < (10 - 2) is %f\n", logic_value ) ;

    logic_value = (2 * 3 < 24 / 3) ;
    printf ( " (2 * 3 < 24 / 3) is %f\n", logic_value ) ;

    logic_value = (10 < 5) ;
    printf ( " (10 < 5) is %f\n", logic_value ) ;

    logic_value = (24 <= 15) ;
    printf ( " (24 <= 15) is %f\n", logic_value ) ;

    logic_value = (36 / 6 <= 2 * 3) ;
    printf ( " (36 / 6 <= 2 * 3) is %f\n", logic_value ) ;

    logic_value = (8 == 8) ;
    printf ( " (8 == 8) is %f\n", logic_value ) ;

    logic_value = (12 + 5 == 15) ;
    printf ( " (12 + 5 == 15) is %f\n", logic_value ) ;

    logic_value = (8 != 5) ;
    printf ( " (8 != 5) is %f\n", logic_value ) ;

    logic_value = (15 != 3 * 5) ;
    printf ( " (15 != 3 * 5) is %f\n", logic_value ) ;
}
```


Logic values of the following relations :

$(3 > 5)$ is 0.000000
 $(5 > 3)$ is 1.000000
 $(3 \geq 5)$ is 0.000000
 $(15 \geq 3 * 5)$ is 1.000000
 $(8 < (10 - 2))$ is 0.000000
 $(2 * 3 < 24 / 3)$ is 1.000000
 $(10 < 5)$ is 0.000000
 $(24 \leq 15)$ is 0.000000
 $(36 / 6 \leq 2 * 3)$ is 1.000000
 $(8 == 8)$ is 1.000000
 $(12 + 5 == 15)$ is 0.000000
 $(8 != 5)$ is 1.000000
 $(15 != 3 * 5)$ is 0.000000

ตัวอย่างโปรแกรมที่ 4.12 เป็นโปรแกรมแสดงการกำหนดค่าให้กับตัวแปร แล้วนำมาพิมพ์ที่จอภาพ

```
#include <stdio.h>

main ()
{
    char a_character ;    /* This declares a character. */
    int an_integer ;      /* This declares an integer. */
    float floating_point ; /* This declares a floating point. */

    a_character = 'a' ;
    an_integer = 15 ;
    floating_point = 27.62 ;

    printf ( "%c is the character. \n", a_character ) ;
    printf ( "%d is the integer. \n", an_integer ) ;
    printf ( "%f is the floating point. \n", floating_point ) ;
}
```

a is the character
 15 is the integer.
 27.620000 is the floating point.

ตัวอย่างโปรแกรมที่ 4.13 เป็นโปรแกรมแสดงการกำหนดค่าให้กับตัวแปรเหมือนกับตัวอย่างโปรแกรมที่ 4.12 แต่จะเขียนไว้ที่ส่วนประกาศตัวแปรและผลลัพธ์ที่ได้จะเหมือนกัน

```
#include <stdio.h>

main ()
{
    char a_character = 'a';          /* This declares / assigns a character. */
    int an_integer = 15;             /* This declares / assigns an integer. */
    float floating_point = 27.62;    /* This declares / assigns a float point. */

    printf ( "%c is the character. \n", a_character );
    printf ( "%d is the integer. \n", an_integer );
    printf ( "%f is the floating point. \n", floating_point );
}
```

ตัวอย่างโปรแกรมที่ 4.14 เป็นโปรแกรมแสดงการใช้ตัวดำเนินการ $+$, $-$, $*$ และ $/$

```
#include <stdio.h>

main ()
{
    float number_1 = 15.0;          /* First arithmetic operator. */
    float number_2 = 3.0;           /* Second arithmetic operator. */
    float addition_answer;           /* Answer to addition problem. */
    float subtraction_answer;        /* Answer to subtraction problem. */
    float multi_answer;              /* Answer to multiplication problem. */
    float division_answer;           /* Answer to division problem. */

    addition_answer = number_1 + number_2;
    subtraction_answer = number_1 - number_2;
    multi_answer = number_1 * number_2;
    division_answer = number_1 / number_2;

    printf ( "15 + 3 = %f\n", addition_answer );
    printf ( "15 - 3 = %f\n", subtraction_answer );
    printf ( "15 * 3 = %f\n", multi_answer );
    printf ( "15 / 3 = %f\n", division_answer );
}
```

```
15 + 3 = 18.000000
15 - 3 = 12.000000
15 * 3 = 45.000000
15 / 3 = 5.000000
```

ตัวอย่างโปรแกรมที่ 4.15 เป็นโปรแกรมแสดงการใช้ตัวดำเนินการแบบย่อ เช่น

สัญลักษณ์	ตัวอย่าง	ความหมาย
<code>+=</code>	<code>X += Y</code>	<code>X = X + Y ;</code>
<code>-=</code>	<code>X -= Y</code>	<code>X = X - Y ;</code>
<code>*=</code>	<code>X *= Y</code>	<code>X = X * Y ;</code>
<code>/=</code>	<code>X /= Y</code>	<code>X = X / Y ;</code>
<code>%=</code>	<code>X %= Y</code>	<code>X = X % Y ;</code>

```
#include <stdio.h>

main ()
{
    int number = 10;    /* Value of number for example. */

    number += 5 ;
    printf ( "Value of number += 5 is %d\n", number ) ;

    number -= 3 ;
    printf ( "Value of number -= 3 is %d\n", number ) ;

    number *= 3 ;
    printf ( "Value of number *= 3 is %d\n", number ) ;

    number /= 5 ;
    printf ( "Value of number /= 5 is %d\n", number ) ;

    number %= 3 ;
    printf ( "Value of number %%= 3 is %d\n", number ) ;
}
```

```
Value of number += 5 is 15
Value of number -= 3 is 12
Value of number *= 3 is 36
Value of number /= 5 is 7
Value of number %= 3 is 1
```

ตัวอย่างโปรแกรมที่ 4.16 เป็นโปรแกรมที่ใช้ตัวดำเนินการตรรกยะ and โดยอาศัยความสัมพันธ์จากตารางที่ 4.5

```
#include <stdio.h>

main ()
{
    float result ;    /* Result of logical expression. */

    result = 0 && 0 ;
    printf( "0 && 0 = %f\n", result ) ;

    result = 0 && 1 ;
    printf( "0 && 1 = %f\n", result ) ;

    result = 1 && 0 ;
    printf( "1 && 0 = %f\n", result ) ;

    result = 1 && 1 ;
    printf( "1 && 1 = %f\n", result ) ;
}
```

```
0 && 0 = 0.000000
0 && 1 = 0.000000
1 && 0 = 0.000000
1 && 1 = 1.000000
```

ตัวอย่างโปรแกรมที่ 4.17 เป็นโปรแกรมที่ใช้ตัวดำเนินการตรรกยะ or โดยอาศัยความสัมพันธ์จากตารางที่ 4.6

```
#include <stdio.h>

main ()
{
    float result ;    /* Result of logical expression. */

    result = 0 || 0 ;
    printf( "0 || 0 = %f\n", result ) ;

    result = 0 || 1 ;
    printf( "0 || 1 = %f\n", result ) ;

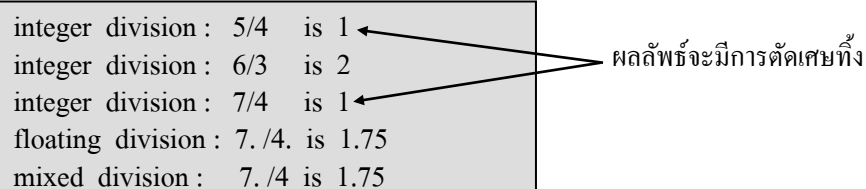
    result = 1 || 0 ;
    printf( "1 || 0 = %f\n", result ) ;

    result = 1 || 1 ;
    printf( "1 || 1 = %f\n", result ) ;
}
```

```
0 || 0 = 0.000000
0 || 1 = 1.000000
1 || 0 = 1.000000
1 || 1 = 1.000000
```

ตัวอย่างโปรแกรมที่ 4.18 เป็นโปรแกรมแสดงการหาร

```
#include <stdio.h>
int main (void)
{
    printf ( "integer division : 5/4    is %d \n", 5/4 );
    printf ( "integer division : 6/3    is %d \n", 6/3 );
    printf ( "integer division : 7/4    is %d \n", 7/4 );
    printf ( "floating division : 7. /4. is %1.2f \n", 7. /4. );
    printf ( "mixed division ; 7. /4    is %1.2f \n", 7. /4 );
    return 0 ;
}
```



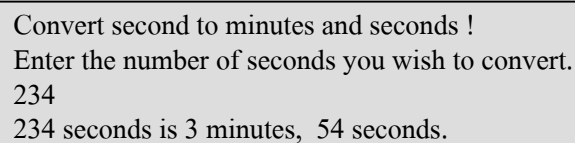
```
integer division : 5/4    is 1
integer division : 6/3    is 2
integer division : 7/4    is 1
floating division : 7. /4. is 1.75
mixed division : 7. /4 is 1.75
```

ผลลัพธ์จะมีการตัดเศษทิ้ง

ตัวอย่างโปรแกรมที่ 4.19 เป็นโปรแกรมการเปลี่ยนหน่วยจากวินาทีเป็นนาทีและวินาที โดยให้ผู้ใช้ป้อนข้อมูลเข้าไปโดยจะอาศัยตัวดำเนินการ / และ %

```
/* Converts seconds to minutes and seconds */
#include <stdio.h>
#define SEC_PER_MIN 60      /* seconds in a minute */
int main (void)
{
    int sec, min, left ;

    printf ( "Convert seconds to minutes and seconds ! \n" );
    printf ( "Enter the number of seconds you wish to convert. \n" );
    scanf ( "%d", &sec);      /* number of seconds is read in */
    min = sec / SEC_PER_MIN ; /* truncated number of minutes */
    left = sec % SEC_PER_MIN ; /* number of seconds left over */
    printf ( "%d seconds is %d minutes, %d seconds. \n", sec, min, left );
    return 0 ;
}
```



```
Convert second to minutes and seconds !
Enter the number of seconds you wish to convert.
234
234 seconds is 3 minutes, 54 seconds.
```

ตัวอย่างโปรแกรมที่ 4.20 เป็นโปรแกรมที่จะมีการแปลงชนิดข้อมูลให้โดยอัตโนมัติ ก่อนที่จะมีการดำเนินการ

```

1  /* Automatic type conversions */
2  #include <stdio.h>
3  int main (void)
4  {
5      char ch ;
6      int i ;
7      float fl ;

8      fl = i = ch = 'A' ;          /* line 8 */
9      printf( "ch = %c, i = %d, fl = %2.2f\n", ch, i, fl ) ;
10     ch = ch + 1 ;                /* line 10 */
11     i = fl + 2 * ch ;            /* line 11 */
12     fl = 2.0 * ch + i ;          /* line 12 */
13     printf( "ch = %c, i = %d, fl = %2.2f\n", ch, i, fl ) ;
14     return 0 ;
}
```

ch = A, i = 65, fl = 65.00 ch = B, i = 197, fl = 329.00
--

จากบรรทัดที่ 8 และ 9 ตัวอักษร 'A' เป็นรหัสแอสกี มีขนาด 1 ไบต์ ถูกเก็บไว้ในตัวแปร ch ตัวแปรจำนวนเต็ม i จะได้รับค่าจำนวนเต็มเป็น 65 ซึ่งตัวอักษร 'A' มีค่าเป็น 65 ซึ่งตัวแปร i จะมีขนาด 2 ไบต์ สุดท้ายตัวแปร fl จะได้รับค่า 65.00 ซึ่งถูกแปลงจากจำนวนเต็มเป็นจำนวนจุดลอยตัวจาก 65

บรรทัดที่ 10 ตัวอักษร 'A' จะถูกแปลงเป็น 65 ก่อน แล้วเพิ่มขึ้น 1 เป็น 66 แล้วเก็บไว้ในตัวอักษร ch และในบรรทัดที่ 13 เราจะนำค่า ch มาพิมพ์ด้วยตัวระบุการแปลงผันแบบ %c จะได้รับรหัสแอสกีเป็น 'B'

บรรทัดที่ 11 ค่าของ ch ขณะนี้มีค่า 66 นำมาคูณกับ 2 จะได้ผลลัพธ์เป็น 132 และต้องถูกแปลงเป็นจำนวนทศนิยมก่อน คือ 132.00 แล้วนำมาบวกกับ fl ซึ่งก่อนหน้านี้มีค่า 65.00 ผลลัพธ์ที่ได้จะเป็น 197.00 แต่ต้องถูกแปลงเป็นจำนวนเต็มก่อน คือ 197 แล้วนำไปเก็บไว้ในตัวแปรจำนวนเต็ม i โดยในบรรทัดที่ 13 เราจะพิมพ์ค่า i โดยใช้ตัวระบุการแปลงผันแบบ %d

บรรทัดที่ 12 นำค่าของ ch ขณะนี้มีค่า 66 ต้องมีการแปลงเป็น 66.00 แล้วนำมาคูณกับ 2.0 ได้ผลลัพธ์เป็น 132.00 และนำมาบวกกับค่า i ซึ่งมีค่าเป็น 197 และจะถูกแปลงเป็น 197.00 แล้วนำมาบวกกันจะได้ผลลัพธ์เป็น 329.00