02204171 Structured Programming

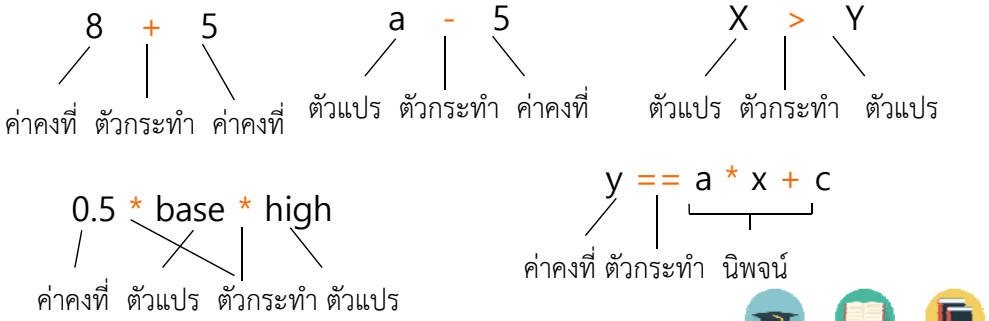
Chapter 3 : C Programming Expression

Outline

- Arithmetic / Mathematics Expression
  - Basic operators
  - Compound Operators
  - Mathematic functions
- Relational Expression
- Logical Expression
- Operator Precedence

นิพจน์ในภาษา C (Expression)

- **นิพจน์ (Expression)** คือ การนำข้อมูลในรูปแบบของค่าคงที่หรือตัวแปรมาเชื่อมต่อกันด้วยเครื่องหมายต่างๆ เพื่อให้ได้ผลลัพธ์ค่าใดค่าหนึ่ง
  - ข้อมูลในรูปแบบของค่าคงที่หรือตัวแปร เรียกว่า **ตัวถูกกระทำ (Operand)**
  - เครื่องหมาย เรียกว่า **ตัวกระทำ หรือ ตัวดำเนินการ (Operator)**



นิพจน์ในภาษา C (Expression)

- นิพจน์คณิตศาสตร์
  Arithmetic / Mathematics Expression
- นิพจน์ความสัมพันธ์
  Relational Expression
- นิพจน์ตรรกศาสตร์
  Logical Expression

1. Arithmetic / Mathematics Expression

- Basic operators
- Compound Operators
- Mathematic functions

1. Arithmetic / Mathematics Expression

- Basic operators

`+` `-` `*` `/` `%`

การหาร แบ่งเป็นการหารแบบจำนวนเต็มและจำนวนจริง

ตัวอย่าง

```
2 + 4.9
3.8 - 4
2 * 3.4
5 / 2.0
5 / 2
5 % 2
5.0 % 2
-5 / 2
-5 % 2
11 % 3
5 / 0
5 % 0
2.5*2
2.5/2
```


1. Arithmetic / Mathematics Expression

- Basic operators

% คือ ตัวดำเนินการหารเอาเศษ

17/5 = 3

```
   3
5 ) 17
   15
    2
```

17%5 = 2

Note: เป็นตัวดำเนินการทางคณิตศาสตร์เพียงตัวเดียวที่กำหนดให้ใช้กับค่าจำนวนเต็ม เท่านั้น

1. Arithmetic / Mathematics Expression

- Basic operators

ลำดับการทำงานของตัวดำเนินการคณิตศาสตร์

1. ( )
2. * / %
3. + -

Note: ถ้าลำดับเท่ากันคำนวณจากซ้ายไปขวา

```
int  Width,High;

Width = 2*5+(6*2)/5;
High= (6+5)+102%2;
```

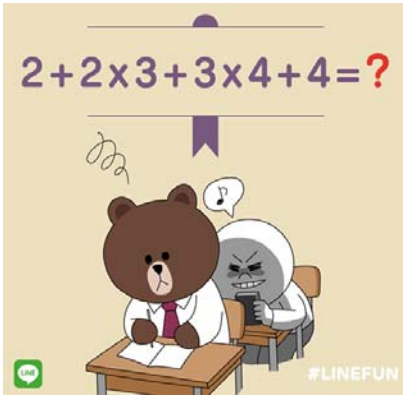
Width **12** (int)

High **11** (int)

1. Arithmetic / Mathematics Expression

- Basic operators

ตัวอย่าง: ลำดับการทำงานของตัวดำเนินการคณิตศาสตร์




```
X = 9*3/2-5%2+7
X = 9*3/2-5%2+7
X = 27/2-5%2+7
X = 27/2-5%2+7
X = 13-5%2+7
X = 13-5%2+7
X = 13-1+7
X = 13-1+7
X = 12+7
X = 19
```


Quick Check 1

- 3/5
- 3/5.0
- 25 % 3
- 14.0 % 2
- 17 + 8 / 5 -7
- 17.0 + 8.0 / 5.0 - 7.0
- 17 + 8 /5.0 - 7.0
- 10 % 3 - 5 / 2
- 10 % (3 – 4) / 2

1. Arithmetic / Mathematics Expression

- Basic operators (Unary operator)

ตัวดำเนินการเพิ่ม และลดค่า (Increment & Decrement)

1. ++ คือ ตัวดำเนินการเพิ่มค่า
   (จะบวกหนึ่งเข้ากับตัวถูกดำเนินการ)
2. -- คือ ตัวดำเนินการลดค่า
   (จะลบหนึ่งเข้ากับตัวถูกดำเนินการ)

```
x++; จะเหมือนกับ x = x+1;
++x; จะเหมือนกับ x = x+1;
```

```
x--; จะเหมือนกับ x = x-1;
--x; จะเหมือนกับ x = x-1;
```


1. Arithmetic / Mathematics Expression

- Basic operators (Unary operator)

ตัวดำเนินการเพิ่ม และลดค่า (Increment & Decrement)

กำหนดให้ x = 5

ตัวดำเนินการแบบมาก่อน (prefix)

n = ++x;
→ x = x+1;
→ n = x;
จะเป็นการกำหนดค่า n ให้กับ 6

n = --x;
→ x = x-1;
→ n = x;
จะเป็นการกำหนดค่า n ให้กับ 4

ตัวดำเนินการแบบตามหลัง (postfix)

n = x++;
→ n = x;
→ x = x+1;
จะเป็นการกำหนดค่า n ให้กับ 5

n = x--;
→ n = x;
→ x = x-1;
จะเป็นการกำหนดค่า n ให้กับ 5

ตัวดำเนินการนี้จะใช้ได้กับเฉพาะ**ตัวแปร**เท่านั้น จะใช้กับนิพจน์อื่นๆ ไม่ได้ เช่น (i+j)++

1. Arithmetic / Mathematics Expression

- Compound Operators

ตัวดำเนินการประกอบ คือ ตัวดำเนินการที่เป็น**รูปแบบย่อ** ของตัวดำเนินการ (Operator) และตัวแปรที่ถูกดำเนินการ (Operand)

1. +=
2. -=
3. *=
4. /=
5. %=

การใช้ตัวดำเนินการประกอบมีรูปแบบดังนี้

```
variable operator= expression;
```

มีความหมายเช่น เดียวกันกับ

```
variable = variable operator expression;
```


1. Arithmetic / Mathematics Expression

- Compound Operators

ตัวอย่างการใช้ตัวดำเนินการ

```
x+=1   ➔ x = x+1
x+=5   ➔ x = x+5
x*=7   ➔ x = x*7
x-=9   ➔ x = x-9
x-=-9  ➔ x = x-(-9)
x/=5   ➔ x = x/5
x%=5   ➔ x = x%5
x%=y   ➔ x = x%y
```

X 7 (int)

1. Arithmetic / Mathematics Expression

- ตัวอย่าง: Unary & Compound Operators

```
int y = 10, z;
z = y--;
printf("z = y--; y = %d, z = %d\n", y, z);

y = 10;
z = --y;
printf("z = --y; y = %d, z = %d\n", y, z);

z*=y+6;
printf("y = %d, z = %d\n", y, z);
```


Quick Check 2

จงแสดงผลลัพธ์ของโปรแกรมต่อไปนี้

```
#include <stdio.h>
int main()
{
   const int x = 0.5;
   float area,width=5, high=3;

   area = x*width*high;
   printf("%.2f\n",  area);

   return 0;
}
```

```
#include <stdio.h>
int main()
{
   float  width=5, high=3;
   int area;

   area = 0.5*width*high;
   printf("%d\n",  area);

   return 0;
}
```

1. Arithmetic / Mathematics Expression

Type Conversion (Casting)

คอมพิวเตอร์ต้องเก็บผลลัพธ์ เป็นชนิดใดชนิดหนึ่ง เมื่อการดำเนินการมีชนิดข้อมูลหลายชนิดปนกันอยู่ โปรแกรมจึงต้องเลือกว่าควรคำนวณในรูปแบบใด **(ชนิดข้อมูลใด)**

- หากไม่มีการระบุชนิดข้อมูลจากผู้เขียนโปรแกรม ภาษาซีจะเลือกชนิดของข้อมูลให้ตามกฎการเปลี่ยนชนิดข้อมูล การเปลี่ยนโดยนัยของกฎแบบนี้เรียกว่า **Implicit Type Conversion**
- ผู้เขียนโปรแกรมสามารถระบุชนิดข้อมูลที่ต้องการเปลี่ยนโดยตรงก็ได้ เรียกว่า **Explicit Type Conversion**

1. Arithmetic / Mathematics Expression

Implicit Type Conversion (Casting)

คือ การที่คอมไพเลอร์เลือกเปลี่ยนชนิดข้อมูลให้อัตโนมัติในการคำนวณ โดยการแปลงชนิดข้อมูลจะแปลงไปสู่ชนิดข้อมูลที่มีนัยสำคัญมากกว่า

ลำดับนัยสำคัญ	ชนิดข้อมูล
1 (มีนัยสำคัญสูงสุด)	double
2	float
3	unsigned int
4	int
5	short
6 (มีนัยสำคัญต่ำสุด)	char

1. Arithmetic / Mathematics Expression

Implicit Type Conversion (Casting)

ตัวอย่าง 1: การแปลงประเภทข้อมูล

```
float myfloat=25.49;
int myint=1;
char mychar='A';

printf ("Char + Int = %d\n" ,mychar+myint);
printf ("Char + Float = %f\n" ,mychar+myfloat);
printf ("Int + Float = %f\n" ,myint+myfloat);
printf ("Int + Float = %d\n" ,myint+myfloat);
printf ("Char + Int = %c\n" ,mychar  + myint);
```

การแปลงข้อมูลกับ printf จะต้องระบุ format code (ชนิดข้อมูลที่จะแสดงใน printf) ให้เหมือนกับชนิดที่คอมไพเลอร์แปลงให้

1. Arithmetic / Mathematics Expression

Explicit Type Conversion (Casting)

ไวยกรณ์การแปลงชนิดข้อมูล (Syntax)

(typename) expression;

average (float)

```
int sum = 60;
int count = 80;
float average;
average = sum / count;
```

average (float)

```
int sum = 60;
int count = 80;
float average;
average = (float) sum/count;
```

Explicit conversion

1. Arithmetic / Mathematics Expression

- Explicit Type Conversion (Casting)

ตัวอย่าง 2: การแปลงประเภทข้อมูล

```
#include <stdio.h>
int main()
{
  const float PI = 22/7.0;
  int r=1,h=3;

  printf("Volume of the cone is %.2f\n", 1/3*PI*r*r*h);
  printf("Volume of the cone is %.2f\n", PI*r*r*h*1/3);

  printf("Volume of the cone is %.2f\n", (float)1/3*PI*r*r*h);
  printf("Volume of the cone is %.2f\n", (float)(1/3)*PI*r*r*h);
  return 0;
}
```


1. Arithmetic / Mathematics Expression

- Explicit Type Conversion (Casting)

ตัวอย่าง 3: การแปลงประเภทข้อมูล

```
float myfloat=25.49;
char mychar='A';

printf ("Char + Float = %f\n" ,mychar+myfloat);
printf ("(char)(Char + Float) = %c\n" ,(char)(mychar+myfloat));
printf ("Char + (char)Float= %d\n" ,mychar+ (char)myfloat);
```


1. Arithmetic /Mathematics Expression

- **Quick check 3** จงหาค่าของตัวแปร a, b, c และ d หลังจากประมวลผลแต่ละนิพจน์ จากส่วนของโปรแกรมต่อไปนี้ (**หมายเหตุ** ทุกนิพจน์ประมวลผลต่อเนื่องกันไปเรื่อยๆ)

const int x=3;
int a=0,b=0,c=0;
float d=0;
	ค่าของตัวแปร			

	a	**b**	**c**	**d**
	0	0	0	0.0
b=1;				
c=x+b;				
a+=x*4;				
d=c;				
a/=d;				
d+=a-b;				
a=b++;				
c=b%5;				

1. Arithmetic / Mathematics Expression

- Mathematic functions

ฟังก์ชั่นในไลบารี่ math.h

ฟังก์ชัน	หน้าที่	
sqrt(x)	ค่ารากที่สองของ x	Power functions
pow(x, y)	x^y	
sin(x)	ค่าไซน์ (sine) ของ x (หน่วยเรเดียน)	Trigonometric functions
cos(x)	ค่าโคไซน์ (cosine) ของ x (หน่วยเรเดียน)	
tan(x)	ค่าแทนเจนต์ (tangent) ของ x (หน่วยเรเดียน)	
exp(x)	e^x	Exponential and logarithmic functions
log(x)	(natural log) ln(x), x > 0	
log10(x)	(log ฐานสิบ) lg(x), x > 0	
ceil(x)	ปัดเลขทศนิยมให้เป็นมีค่าเป็นจำนวนเต็มน้อยสุดที่ >= x	Rounding functions
floor(x)	ปัดเลขทศนิยมให้เป็นมีค่าเป็นจำนวนเต็มมากสุดที่ <= x	
abs(x)	ค่า absolute |x|	

1. Arithmetic / Mathematics Expression

- Mathematic functions

ตัวอย่าง

$\sqrt{5x}$

```
sqrt(x*5)
```

$\left\lfloor \frac{x}{y} \right\rfloor$

```
floor(x/y)
```

$(x+y)^z$

```
pow(x+y,z)
```

$\lceil x-y \rceil$

```
ceil(x-y)
```


1. Arithmetic / Mathematics Expression

- Mathematic functions

ตัวอย่าง: คำนวณปริมาตรทรงกลม

```
#include <stdio.h>
#include <math.h>
int main()
{
    float vol, radius;
    scanf ("%f",&radius);

    vol = 4 * 3.1416 * pow(radius,3)/3;

    printf("volume of the sphere %.2f",vol);

    return 0;
}
```


2. Relational Expression

ตัวดำเนินการสัมพันธ์ ใช้ในการเปรียบเทียบและตัดสินใจ ซึ่งผลของการเปรียบเทียบจะเป็นได้ 2 กรณีเท่านั้นคือ **จริง** หรือ **เท็จ**

==	เท่ากัน (Equality)
!=	ไม่เท่ากัน (Inequality)
<	น้อยกว่า (Less Than)
<=	น้อยกว่าหรือเท่ากัน (Less Than or Equal)
>	มากกว่า (Greater Than)
>=	มากกว่าหรือเท่ากัน (Greater Than or Equal)

2. Relational Expression

ตัวอย่าง 1

- 5 == 5
- 5 != 5
- 5 < 5.5
- 5 <= 5
- 10 > 50

ผลลัพธ์ของนิพจน์ความสัมพันธ์มีค่าเป็นจำนวนเต็ม และมีค่าได้เพียงสองค่า คือ

- 1 หรือตัวเลขใดๆ แทนค่าความจริงเป็นจริง
- 0 แทนค่าความจริงเป็นเท็จ

ตัวอย่าง 2

- 'A' > 'F'
- 'a' < 'Z'
- '5' > '1'
- 5 > 100
- 5 < '1'

('A' = 65 และ 'F' = 70)

('a' = 97 และ 'Z' = 90)

('5' = 53 และ '1' = 49)

Fun Fact: ASCII code

- แอสกี (ASCII) ย่อมาจาก “American Standard Code for Information Interchange” รหัสนี้แทนตัวอักษรได้ **128 ตัว** ถูกสร้างขึ้นราวช่วงทศวรรษที่ 1960s และเป็นรหัสแทนตัวอักษรที่นิยมใช้กันมากที่สุด

- นอกจากนี้ยังมีอีก 128 ตัว สำหรับ extended ASCII code แทนตัวอักษรกรอบ พื้น ลายเส้น และตัวอักษรกรีก-ละติน

ASCII control characters

DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de linea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

ASCII printable characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	|
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_			

theASCIIcode.com.ar

3. Logical Expression

นิพจน์ตรรกศาสตร์ ใช้ในประมวลผลทางตรรกะกับค่าตัวเลข หรือค่าของตัวแปร หรือนิพจน์ ผลลัพธ์ที่ได้ จะมีเพียง 2 ค่าคือ **จริง** หรือ **เท็จ** (เช่นเดียวกับผลลัพธ์ของนิพจน์เชิงสัมพันธ์)

ตัวดำเนินการ		ตัวอย่าง	
!	นิเสธ (NOT)	!(5 == 5)	0
&&	และ (AND)	5 < 6 && 6 < 10	1
||	หรือ (OR)	5 < 6 || 6 < -1	1

```
! 0 = 1
! 1 = 0
```

```
0 && 0 = 0
0 && 1 = 0
1 && 0 = 0
1 && 1 = 1
```

```
0 || 0 = 0
0 || 1 = 1
1 || 0 = 1
1 || 1 = 1
```

นิเสธ คือการกลับค่าความจริงของตัวถูกดำเนินการที่ตามหลังเครื่องหมาย

3. Logical Expression

ตัวอย่าง 1

- !20
- 10 && 5
- 0.1 || 0.0
- 10 && 0.0
- 0 || 0
- !0
- !!!0
- !!-5
- 0 || 5
- !10 && 5

ตัวอย่าง 2

- ! (-1 || 0)
- ! (1 || 9 && 0)
- ! ((1 || 9) && 0)
- ! ((1 || 0) && 0)
- 8 && (a - 6/2)

4. Operator Precedence

ลำดับการทำงาน	เครื่องหมาย	การทำงานกรณีลำดับเดียวกัน
1	()	โดยทำจากซ้ายไปขวา
2	!, ++, --	โดยทำจากขวาไปซ้าย
3	* , / , %	โดยทำจากซ้ายไปขวา
4	+ , -	โดยทำจากซ้ายไปขวา
5	< , <= , > , >=	โดยทำจากซ้ายไปขวา
6	== , !=	โดยทำจากซ้ายไปขวา
7	&&	โดยทำจากซ้ายไปขวา
8	||	โดยทำจากซ้ายไปขวา
9	= +=, -=, *=, /=, %=	โดยทำจากขวาไปซ้าย

4. Operator Precedence

ตัวอย่าง 1

```
#include <stdio.h>
int main()
{
 int a=1, b=2, c=3, n;
 n = ++a  * b  -  c-- ;
 printf("a = %d, b = %d and c = %d\n",a,b,c);
 printf("with n = %d", n);
 return 0;
}
```


4. Operator Precedence

ตัวอย่าง 2

```
#include <stdio.h>
int main()
{
  int x=5,y=10,z=3;
  x*= y < 3*z++;

  printf("%d\n",x);
  return 0;
}
```

```
#include <stdio.h>
int main()
{
  int x=5,y=10,z=3;
  x*= y < 3*++z;

  printf("%d\n",x);
  return 0;
}
```


Quick Check 4

- จงหาผลลัพธ์ของนิพจน์ต่อไปนี้

x	y	Expression		Result
12	2	x+3 <= y*10		
20	2	x-3 <= y*10		
7	1	x+3 != y*10		
17	2	x+3 == y*10		
100	5	x+3 > y*10		