#2 Operator

Operator หรือตัวดำเนินการของภาษา C โดยจะมี Operator 2 แบบใหญ่ๆ

- **Unary Operator** คือเป็นตัวดำเนินการที่ดำเนินการกับ Operand 1 ตัว
- **Binary Operator** คือเป็นตัวดำเนินการที่ดำเนินการกับ Operand 2 ตัว

ซึ่งก็จะแบ่งย่อยลงไปอีก ซึ่งแต่ละ Operator ก็จะมีการเรียงความสำคัญที่ต่างกันดังตาราง

Precedence	Operator	Description	Associativity
1	++ -- ( ) [ ] . –> (*type*){*list*}	Suffix/postfix increment and decrement Function call Array subscripting Structure and union member access Structure and union member access through pointer Compound literal(C99)	Left-to-right
2	++ -- + – ! ~ (*type*) * & sizeof _Alignof	Prefix increment and decrement Unary plus and minus Logical NOT and bitwise NOT Type cast Indirection (dereference) Address-of Size-of Alignment requirement(C11)	Right-to-left
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ –	Addition and subtraction	
5	<< >>	Bitwise left shift and right shift	
6	< <= > >=	For relational operators < and ≤ respectively For relational operators > and ≥ respectively	
7	== !=	For relational = and ≠ respectively	
8	&	Bitwise AND	
9	^	Bitwise XOR (exclusive or)	
10	|	Bitwise OR (inclusive or)	
11	&&	Logical AND	
12	||	Logical OR	
13	?:	Ternary conditional	Right-to-Left
14	= += –= *= /= %= <<= >>= &= ^= |=	Simple assignment Assignment by sum and difference Assignment by product, quotient, and remainder Assignment by bitwise left shift and right shift Assignment by bitwise AND, XOR, and OR	
15	,	Comma	Left-to-right

ตัวที่อยู่สูงกว่า (Precedence สูงกว่า) โปรแกรมก็จะทำ Operator นั้นๆก่อนตัวอื่น แล้ว Operator ยังมีคุณสมบัติในการที่จะ Chain กันได้อีก เช่น

```
#include <stdio.h> int main() { int x; int y; x = 5 + 2 * 2; y = x = 10;
// y = ? }
```

Arithmetic Operator

ตัวดำเนินการทางคณิตศาสตร์จะประกอบด้วย

Operator	Description
+	Addition
-	Subtracts
*	Multiplication
/	Division
%	Modulo
++	Increase
--	Decrease

Relational Operators

ตัวดำเนินการที่เอาไว้เปรียบเทียบ

Operator	Description
==	Equality
!=	Inequality
>	Greater
>=	Greater or equal to
<	Less than
<=	Less than or equal to

Logical Operators

ตัวดำเนินการเปรียบเทียบทาง Logical

Operator	Description
&&	Logical AND
||	Logical OR
!	Logical NOT

Bitwise Operators

ตัวเดินการกับ Bit

Operator	Description
&	Binary AND
|	Binary OR
^	Binary XOR
~	Binary One's compliment
<<	Binary Left Shift
>>	Binary Right Shift

Assignment Operators

เป็น Operator ที่ใช้สำหรับการกำหนดค่าให้กับตัวแปร

Operator	Description
=	Assignment
+=	Add and assignment
-=	Subtract and assignment
*=	Multiply and assignment
/=	Divide and assignment
%=	Modulus and assignment
<<=	Left SHIFT and assignment
>>=	Right SHIFT and assignment
&=	Bitwise AND and assignment
|=	Bitwise OR and assignment
^=	Bitwise Exclusive OR and assignment

Misc Operators

Operator	Description
sizeof()	Return size of a variable or type
&	Return the address of variable
*	Pointer to a variable
? :	Condition Expression [Condition ? true : false]
,	Comma Operator (Connect expression)

Type Conversion

Explicit Type Conversion

เราสามารถ Casting Type ของตัวแปรได้ (แต่ไม่ได้เขียนทับค่าของตัวแปรนั้นๆ) โดยการใช้

```
(type) variable
```

ตัวอย่างเช่นเราต้องการให้ตัวแปร Float กลายเป็น Int

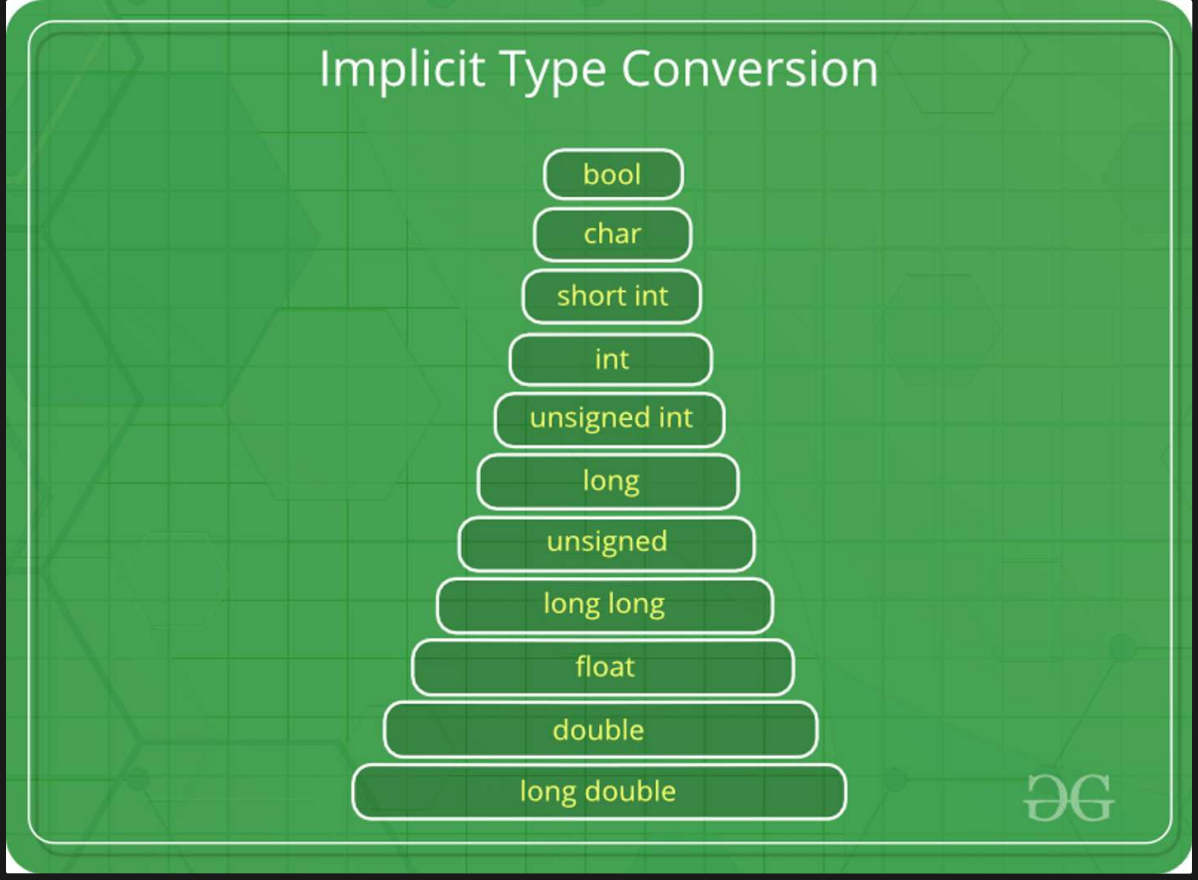
```
#include <stdio.h> int main() { float x = 2.95; printf("Result: %d", (int) x); }
```

```
Result: 2
```

จะสังเกตว่าจะไม่มีการปัดเศษขึ้นแต่จะเป็นการตัดเศษทิ้งเลย

Implicit Type Conversion

ในภาษา C การที่นำตัวแปรมากระทำกันผ่าน Operator Compiler จะทำการแปลง Type ที่ Narrow -> Wider เพื่อไม่ใช้เสียข้อมูลไป



แหล่งที่มา

GeeksForGeek

ตัวอย่างการแปลง

Character → integer

boolean → integer

เช่น

```
// An example of implicit conversion #include <stdio.h> int main() { int x = 10; // integer x char y = 'a'; // character c // y implicitly converted to int. ASCII // value of 'a' is 97 x = x + y; // 10 + 97 // x is implicitly converted to float float z = x + 1.0; printf("x = %d, z = %f", x, z); // x = 107, z = 108.000000 return 0; }
```

More Example

ในบางกรณีเราอาจจะใช้ใน Expression ที่กระทำกันแบบคนละ Type ลองพิจารณา Code ข้างล่าง

```
#include <stdio.h> int main() { int a = 5; int b = 1; printf("Result: %f", b / a); }
```