

---

# C Programming & Lab

## 4. Expressions

Sejong University

---

# Outline

---

- 1) **Equations and Operators**
- 2) **Arithmetic Operators**
- 3) **Assignment Operators**
  - 3-1) **Increment/Decrement Operators**
- 4) **Relational Operators**
- 5) **Logical Operators**
- 6) **Conditional Operators**
- 7) **Precedence and Associativity of Operator**

# 1) Expression and Operator

---

- **Operator**

- Basic syntax in programming
- Arithmetic, Assignment, Relational, Logical, Bitwise, Increment/Decrement, Conditional operators.

- **Expression**

- Any combination of operands and operators

<code>int num = 10;</code>	A constant 10
<code>int num1 = num;</code>	A variable num
<code>int num2 = num + 1;</code>	An operation expression num+1
<code>int num3 = strlen("abc");</code>	A function <code>strlen("abc")</code> , returning a value

- **Operation expression**

- Consists of operands and operators

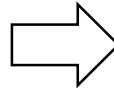
## 2) Arithmetic Operator

---

- **Arithmetic Operator**
  - +, -, \*, /, % (modulus)
  - Contains two operands
  - Operates from left to right
- **Addition/Subtraction**
  - A+B is A plus B, A-B is A minus B

Source code

```
int A = 3;  
int B = 6;  
printf("%d\n", A+B);  
printf("%d\n", A-B);
```



Display

```
9  
-3
```

## 2) Arithmetic Operator

---

- **Multiplication/Division**

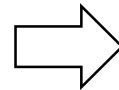
- $A * B$  is A is multiplied by B,  $A / B$  is A is divided by B
- In division, data type matters! Results may change!

- **Modulus**

- $A \% B$  is to calculate the remainder of  $A / B$ .
- Only integer type
  - ✓ For integer type, / and % compute quotient and remainder

Source code

```
int A = 10;  
int B = 3;  
printf("%d\n", A * B);  
printf("%d\n", A / B);  
printf("%d\n", A % B);
```



Display

```
30  
3  
1
```

## 2) Arithmetic Operator

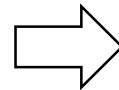
---

- **Data Type Conversion**

- Using different data types, converted to the data type that is wider than the other
  - ✓ Integer and Integer → Integer
  - ✓ Real number and Real number → Real number
  - ✓ Real number and Integer → Real number

Source code

```
printf("%f\n", 3+4.0f);  
printf("%f\n", 10/3.0f);
```



Display

```
7.000000  
3.333333
```

## 2) Arithmetic Operator

---

- **Example**

- What will be the result of the following source code?

```
printf("%d\n", 3*4);  
printf("%f\n", 2.0f*3.0f);  
printf("%f\n", 2.0f*4);  
printf("%d\n", 10/2);  
printf("%d\n", 10/3);  
printf("%f\n", 10/3.0f);  
printf("%d\n", 10%3);  
printf("%f\n", 3.0f+3/2);
```

## 2) Arithmetic Operator

---

- **Example**
  - Result

```
12
6.000000
8.000000
5
3
3.333333
1
4.000000
```



## 2) Arithmetic Operator

---

- **Operator Precedence**

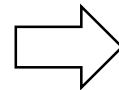
- Follow the precedence of operators when multiple operators are used together
- Precedence: Multiplication/Division/Modulus > Addition/Subtraction > Assignment

- **Parentheses Operator**

- Can manipulate the order of operations

Source code

```
printf("%d\n", 3+6*3);  
printf("%d\n", (3+6)*3);  
printf("%f\n", 3.0f+3/2);  
printf("%f\n", (3.0f+3)/2);  
printf("%f\n", 3.0f/(3+2));
```



Display

```
21  
27  
4.000000  
3.000000  
0.600000
```

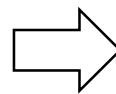
## 2) Arithmetic Operator

---

- **Up-casting (Implicit type conversion)**
  - In assignment operation, the value on the right side is converted to the data type of the variable on the left side
  - In expression operation, convert to the data type that is wider than the other
- **Down-casting (Explicit type conversion)**
  - (data type)A explicitly converts the data type of A to the denoted data type
  - $\text{int} \rightarrow \text{float}$ ,  $\text{float} \rightarrow \text{int}$ ,  $\text{int} \rightarrow \text{char}$
  - Warning! may lose information

Source code

```
printf("%d\n", (int)3.81f);
```



Display

3

## 2) Arithmetic Operator

---

- **Example**

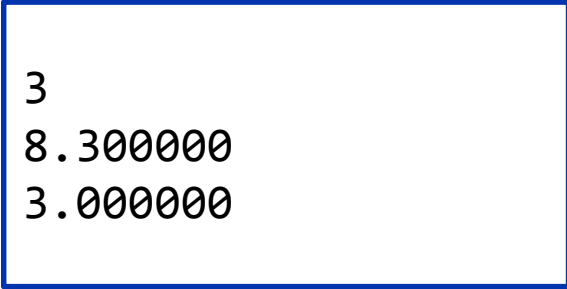
- What will be the result of the following source code?
- Run the code in C

```
int a;  
a = 3.3f;  
printf("%d\n", a);  
printf("%f\n", 3.3f+5);  
printf("%f\n", (float)a);
```

## 2) Arithmetic Operator

---

- **Example**
  - Display




3  
8.300000  
3.000000

### 3) Assignment Operator

---

- **Assignment Operator**

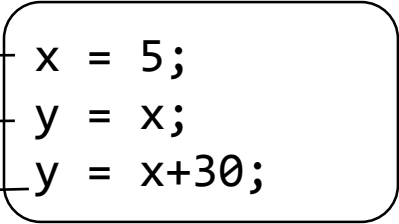
- Differ from “equals sign” in mathematics
- The value on the right is stored in the memory denoted by a variable on the left side
- Pre-stored value is overwritten by a new value



**Variable = Value**

- **Different types of assignment**

- Variable= Value; ←
- Variable= Variable; ←
- Variable= Expression; ←



```
x = 5;  
y = x;  
y = x+30;
```

✓ 200 = x; // Compilation Error  
x+2 = 0; // Compilation Error

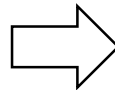
### 3) Assignment Operator

---

- **Statement that assigns a value to a variable**
  - `a = a + 1;`
  - Does not mean that "a and a+1 are the same"
  - Does mean that "Assign the value of a+1 to a variable a ( $a \leftarrow a+1$ )"
- **Assignment Operation Procedure**
  - Assume a variable a contains 20  
`a = a+1;    // a = (pre-stored in a)20 + 1`

Source code

```
a = 20;  
a = a+1;  
printf("%d",a);
```



Display

21

### 3) Assignment Operator

---

- **Cascading Assignment**

- `a = b = c = 3;` This statement indicates that

`c = 3;` //Execute from the right side

`b = c;`

`a = b;`

✓ `a = b+2 = c = d = 5` //Compilation error

`b+2 = c ( X )`

### 3) Assignment Operator

- **Compound Assignment Operator**

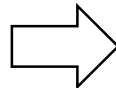
- Compound assignment and arithmetic operators
- Can simplify the source code

Source code

```
int a = 3;
int b = 2;
a += 5;
printf("%d\n", a);
a /= b;
printf("%d\n", a);
a %= 3;
printf("%d\n", a);
```

Compound assignment operator	Meaning
$x += y$	$x = x + y$
$x -= y$	$x = x - y$
$x *= y$	$x = x * y$
$x /= y$	$x = x / y$
$x \%= y$	$x = x \% y$

Display



```
8
4
1
```



## 3-1) Increment/Decrement Operator

- **Increment/Decrement Operator**

- ++ or -- : increase or decrease the value of a variable by 1
- Unary operator
- Pre- and post-increment/decrement are interpreted differently

Operator	Interpretation
++x	Increase x by 1 and do other operations
x++	Do other operations and increase x by 1
--x	Decrease x by 1 and do other operations
x--	Do other operations and decrease x by 1

```
int x = 5;  
int y = x++;  
printf("%d", x);  
printf("%d", y);
```

```
int x = 5;  
int y = --x;  
printf("%d", x);  
printf("%d", y);
```

## 4) Relational Operator

- **Relational Operator**

- Relationship between left side and right side
- Result is always either True or False
- True returns 1 and False returns 0
- In C, any value other than 0 is considered to be true

operator	Interpretation
<	Left side is smaller than right side
<=	Left side is smaller or equal to right side
==	Left side is equal to right side
>=	Left side is greater than or equal to right side.
>	Left side is bigger than right side
!=	Left side and right side are not the same

```
int x = 5;  
int y = 6;  
printf("%d", x>y);  
printf("%d", x<=y);
```

## 4) Relational Operator

---

- **Example**

- What will be the result of the following source code
- Run the code in C

```
int a = 3;
printf("%d\n", a > 4);
printf("%d\n", a < 4);
printf("%d\n", a == 5);
printf("%d\n", a != 3);
printf("%d\n", 2 >= a);
printf("%d\n", a <= a+1);
```

## 4) Relational Operator

---

- **Example**

- Display



0  
1  
0  
0  
0  
1

## 5) Logical Operator

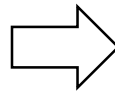
- **Logical Operator**

- To compare one or more conditions and determine whether they are true or false

Operator	Meaning
!	NOT
&&	AND
	OR

Source code

```
int a = 1, b = 0;
printf("%d\n", !a);
printf("%d\n", a&&a);
printf("%d\n", a&&b);
printf("%d\n", a||b);
printf("%d\n", b&&b);
```



Display

```
0
1
0
1
0
```

## 5) Logical Operator

---

- **Example**

- What will be the result of the following source code
- Run the code in C

```
int a = 3;
int b = 5;

printf("%d\n", (a>=3)&&(b<6));
printf("%d\n", (a!=3)&&(a>2));
printf("%d\n", (b!=5)|| (a==1));
printf("%d\n", (a!=!b)|| (b==2));
```

## 5) Logical Operator

---

- **Exmple**
  - Display



1  
0  
0  
1

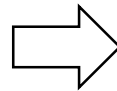
## 6) Conditional Operator

- **Conditional Operator**

- Can replace if ~ else statement
- 3 operands are used: ternary operator

Condition ? A : B	If Condition is true, return A
	If Condition is false, return B

```
int a = 10;  
int b = 5;  
int c;  
a < b ? c = a : c = b;  
printf("%d", c);
```



Display

5



## 7) Precedence and Associativity of Operator

---

- **Operator Precedence**

- If use multiple operators: follow the precedence of operators
- Parentheses can manipulate the order of operators

- **Associativity of Operator**

- Order of operations
- If equal precedence, determined by the associativity of operator
- 예)  $5 / 2 * 4$  Result?
  - ✓ Equal precedence:  $/$  and  $*$
  - ✓ Associativity of operators ( $/$  and  $*$ ): Left to Right

## 7) Precedence and Associativity of Operator

Precedence	Operator	Associativity
1	( ) [ ] . ->	Left to Right
2	*(indirection) & ! ++ --	Right to Left
3	*(multiplication) % /	Left to Right
4	+ -	Left to Right
5	<< >>	Left to Right
6	< > <= >=	Left to Right
7	== !=	Left to Right
8	&	Left to Right
9	^	Left to Right
10		Left to Right
11	&&	Left to Right
12		Left to Right
13	? :	Right to Left
14	= += -= *= %= /= ^= <<= >>=	Right to Left
15	,	Left to Right