Unit 3

Operators and Expressions

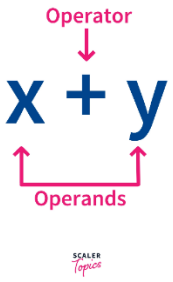
- Operators, Operand, Operation, Expression
- Types of Operators
    - Unary
    - Binary
    - Ternary
    - Arithmetic
    - Relational
    - Logical
    - Assignment
    - Increment/Decrement
    - Conditional
    - Bitwise
    - Size-of Operators

Operators

- An operator is a symbol that operates on a value or a variable. For example: + is an operator to perform addition.

Operand:

- Operand is a value or variable that an operator works on.
- For example, 4+5, here, numbers 4 and 5 are operands whereas + is an operator.



Operation

The action or calculation that an operator performs on its operand(s) to produce a result.

Expression

- A combination of values, variables, operators, and function calls that produces a single value.
- Example: 5 + 3, x > 0 && y < 10 etc.

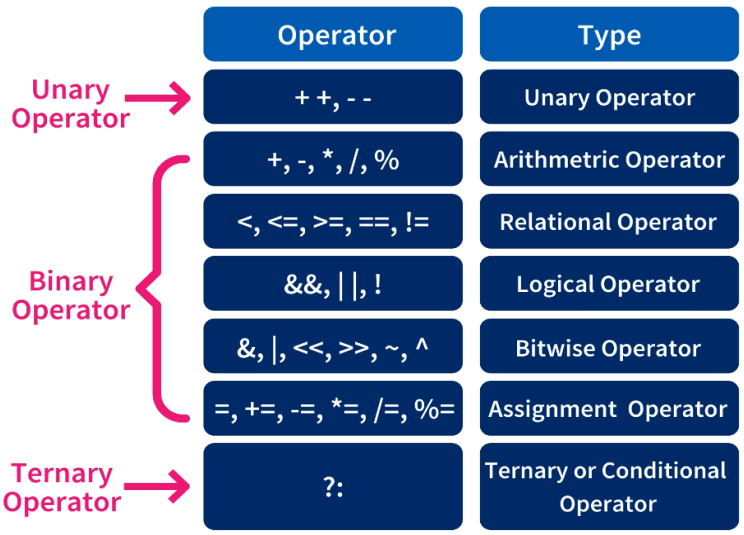
Statement

- A statement is a line of code that performs a specific action or task
- Tasks such as declaring a variable, assigning a value to a variable, or executing a function are statements.
- A statement typically ends with a semicolon (;) in most programming languages.
- Example:
    - int x;
    - x = 5;
    - printf("Hello, world!");

Types of operators in C

Based on number of operands, operators are divided into 3 types.

1. Unary Operator
2. Binary Operator
3. Ternary Operator



SCALER Topics

1. Unary Operator

- Unary operators only operate on single operand.
- There are several unary operators in C programming:

a. Unary plus operator (+): indicates a positive value of the operand, but doesn't change it. For example, +5 returns 5.

b. Unary minus operator (-): negates the operand value. For example, -5 returns -5.

c. Increment operator (++): increases the operand value by 1.

- Pre-Increment Operator: Increments the value before evaluation.
- Post-Increment Operator: Increments the value after evaluation.

Example:

```
int x = 5;
int y, z;

y = ++x; // pre-increment, x is now 6, y is 6
z = x++; // post-increment, x is now 7, z is 6
```

- Decrement operator (--): decreases the operand value by 1. For example, --x on x=5 returns 4.
    - Pre-Decrement Operator: Decrements the value before evaluation.
    - Post-Decrement Operator: Decrements the value after evaluation.

```c
int x = 5;
int y, z;

y = --x; // pre-decrement, x is now 4, y is 4
z = x--; // post-decrement, x is now 3, z is 4
```

2. Binary Operator

- Binary operator operates on two operators.
- There are several types of binary operators:

a. Arithmetic Operator

Arithmetic operators are used to perform mathematical operations on numeric values.

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y

Example

```c
#include <stdio.h>

int main()
{
    int x = 5;
    int y = 3;

    int sum = x + y;
    int difference = x - y;
    int product = x * y;
```

```
    int quotient = x / y;
    int remainder = x % y;

    printf("Sum: %d\n", sum);
    printf("Difference: %d\n", difference);
    printf("Product: %d\n", product);
    printf("Quotient: %d\n", quotient);
    printf("Remainder: %d\n", remainder);

    return 0;
}
```

b. Logical Operator

- Logical operators are used to check logical conditions.
- They return the 1 when the result is true and 0 when the result is false.

Operator	Description	Example (a and b, where a = 1 and b = 0)
&&	Logical AND	a && b, returns 0
||	Logical OR	a || b, returns 1
!	Logical NOT	!a, returns 0

- With AND operator, only if both operands are true, the result is true.
- With the OR operator, if a single operand is true, then the result will be true.
- The NOT operator changes true to false, and false to true.

Example

```
#include <stdio.h>

int main()
{

    int a = 1, b = 0, result;

    // And
    result = (a && b);
    printf("a && b = %d \n", result);

    // Or
    result = (a || b);
```

```
    printf("a || b = %d \n", result);

    // Not
    result = !a;
    printf("!a = %d \n", result);

    return 0;
}
```

c. Relational/Comparison Operator

- Relational/Comparison operators are used to compare two values (or variables).
- If the relation is true, it returns 1; if the relation is false, it returns value 0.

Operator	Meaning of Operator	Example
==	Equal to	5 == 3 is evaluated to 0
>	Greater than	5 > 3 is evaluated to 1
<	Less than	5 < 3 is evaluated to 0
!=	Not equal to	5 != 3 is evaluated to 1
>=	Greater than or equal to	5 >= 3 is evaluated to 1
<=	Less than or equal to	5 <= 3 is evaluated to 0

Example

```
#include <stdio.h>

int main()
{

    int a = 10, b = 20, result;

    // Equal
    result = (a == b);
    printf("(a == b) = %d \n", result);

    // less than
    result = (a < b);
    printf("(a < b) = %d \n", result);
```

```
    // greater than
    result = (a > b);
    printf("(a > b) = %d \n", result);

    // less than equal to
    result = (a <= b);
    printf("(a <= b) = %d \n", result);

    return 0;
}
```

d. Assignment Operator

- The assignment operators are used to assign value to a variable.
- For example: num = 6 will assign the value 6 to the variable num.

Operator	Example	Same as
=	a = b	a = b
+=	a += b	a = a+b
-=	a -= b	a = a-b
*=	a *= b	a = a*b
/=	a /= b	a = a/b
%=	a %= b	a = a%b

Example:

```
#include <stdio.h>

int main()
{

    int a = 10;

    // Assign
    int result = a;
    printf("result = %d \n", result);

    // += operator
    result += a;
    printf("result = %d \n", result);
```

```
    // -= operator
    result -= a;
    printf("result = %d \n", result);

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

    return 0;
}
```

e. Bitwise Operator

- Bitwise operators perform manipulations of data at the bit level.
- These operators also perform the **shifting of bits from right to left**. Bitwise operators are not applied to float or double, long double, void, etc.

Operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise Exclusive OR (XOR)
~	One's complement (NOT)
>>	Shift right
<<	Shift left

Example:

```
#include <stdio.h>

int main()
{

    int a = 0001000, b = 2, result;

    // <<
    result = a << b;
    printf("a << b = %d \n", result);

    // >>
    result = a >> b;
    printf("a >> b = %d \n", result);
```

```
    return 0;
}
```

3. Ternary Operator

- Ternary operator operates on 3 values or variables.
- It is also known as conditional operator.

Syntax: **(Expression1)? Expression2 : Expression3;**

- If (expression 1) returns **true** then the (expression 2) is executed.
- If (expression 1) returns **false** then the expression on the right side of : i.e (expression 3) is executed.

Example

```c
#include <stdio.h>

int main()
{
    int x = 10;
    int y = 5;
    int max = (x > y) ? x : y;

    printf("The maximum value is %d\n", max);

    return 0;
}
```

sizeof() operator

- In C, the sizeof operator is used to determine the size of a variable or data type in bytes.
- Syntax: sizeof(variable)

Example:

```c
#include <stdio.h>

int main()
{
    int num;
```

```c
    float f;
    char c;
    double d;

    printf("The size of an int is %lu bytes\n", sizeof(num));
    printf("The size of a float is %lu bytes\n", sizeof(f));
    printf("The size of a char is %lu bytes\n", sizeof(c));
    printf("The size of a double is %lu bytes\n", sizeof(d));

    return 0;
}
```