Unit-3
Operators and Expressions
(5 marks)

CTEVT Diploma in Computer Engineering
Subject : C Programming
Shree Sarwajanik Secondary Technical School
Prepared By: Er. Abinash Adhikari

1. Operators, Operands, Operation, Expression

- **Operator**
  - An operator is a symbol that performs a specific operation on one or more operands.
  - Example: +, -, *, /, =.
  - In the expression a + b, + is the operator.
- **Operand**
  - An operand is the data or variable on which the operator acts.
  - Example: In a + b, a and b are operands.
- **Operation**
  - An operation is the process performed by the operator on operands.
  - Example: Addition in a + b.
- **Expression**
  - An expression is a combination of operators and operands that computes a value.
  - Example: a + b - c is an expression.

2. Types of Operators

1. Unary Operators

Definition: Operate on a single operand.

Examples:

- Unary minus: -a (negates the value of a).
- Increment: ++a (pre-increment), a++ (post-increment).
- Decrement: --a, a--.
- Logical NOT: !a (negates a Boolean value).]
- Use Cases: Toggle values, adjust counters in loops

Example:

```
int a = 5;

printf("%d", -a); // Output: -5
```

2. Binary Operators

Operate on two operands.

Types:

- Arithmetic Operators:
  - Addition (+), subtraction (-), multiplication (*), division (/), modulus (%).
  - Example:
  - int a = 10, b = 3;
  - printf("%d", a % b); // Output: 1

- Relational Operators:
  - Compare two values and return a Boolean.
  - Examples: >, <, >=, <=, ==, !=.

```
int a = 5, b = 10;
printf("%d", a < b);  // Output: 1 (true)
```

- Logical Operators:
  - Combine multiple conditions.
  - Examples: && (AND), || (OR).

```
int a = 5, b = 10;
printf("%d", (a < b) && (b > 0));  // Output: 1
```

- Bitwise Operators:
  - Work at the bit level.
  - Examples: &, |, ^, ~, <<, >>.

```
int a = 5;  // Binary: 0101
int b = 3;  // Binary: 0011
printf("%d", a & b);  // Output: 1 (Binary: 0001)
```

3. Ternary Operator

- A conditional operator that chooses between two values based on a condition.
- Syntax: condition ? value_if_true : value_if_false.
- Example:

```
int a = 5, b = 10;

int max = (a > b) ? a : b;

printf("%d", max);  // Output: 10
```

4. Assignment Operators

- Used to assign or update values.
- Examples:
  - =: Assigns a value.
  - +=, -=, *=, /=, %=: Shorthand for updating a variable.
- Code Example:

```
int a = 10;

a += 5;  // Equivalent to a = a + 5

printf("%d", a);  // Output: 15
```

5. Increment/Decrement Operators

- Pre-increment/Pre-decrement: Value changes before use (++a, --a).
- Post-increment/Post-decrement: Value changes after use (a++, a--).

- Example:

```
int a = 5;

printf("%d", a++);  // Output: 5

printf("%d", a);    // Output: 6
```

6. Conditional Operator

- Simplifies if-else statements.
- Syntax: condition ? value_if_true : value_if_false.
- Example:

```
int age = 18;

printf("%s", (age >= 18) ? "Adult" : "Minor”);
```

7. Bitwise Operators

- Work on the binary representation of numbers.
- Examples:

  AND (&), OR (|), XOR (^), NOT (~).

  Left shift (<<), right shift (>>).

- Example:

```
int a = 5;  // Binary: 0101

printf("%d", a << 1);  // Output: 10 (Binary: 1010)
```

8. Sizeof Operator

- Determines the size of a data type or variable in bytes.
- Example:

```
printf("%lu", sizeof(int));  // Output: 4 (typically, but may vary by system)
```

operators.c

```c
#include <stdio.h>

int main() {
    int a = 10, b = 5, result;

    // Arithmetic operators
    result = a + b;
    printf("Addition: %d\n", result);

    // Relational operator
    printf("Is a greater than b? %d\n", a > b);

    // Logical operator
    printf("Is a > 0 AND b > 0? %d\n", (a > 0) && (b > 0));

    // Assignment operator
    a += 5;  // a = a + 5
    printf("New value of a: %d\n", a);

    // Increment and decrement
    printf("Post-increment of a: %d\n", a++);  // Outputs original value
    printf("Pre-increment of a: %d\n", ++a);  // Outputs incremented value

    // Conditional operator
    int max = (a > b) ? a : b;
    printf("Maximum value: %d\n", max);

    return 0;
}
```

Output:

```
Addition: 15
Is a greater than b? 1
Is a > 0 AND b > 0? 1
New value of a: 15
Post-increment of a: 15
Pre-increment of a: 17
Maximum value: 17
```