Session 2: Operators and I/O in C Programming

1. Title + Big Idea (The Setup)

Topic Name: Operators and I/O in C Programming

One-line purpose: To understand how to perform operations on data and interact with the user through input and output functions in C.

2. Why Do We Need It? (The Problem)

Problem it solves: Without operators, we can't manipulate data (e.g., add numbers, compare values). Without I/O functions, programs can't receive data from the user or display results, making them isolated and non-interactive.

Real-life analogy: Operators are like the tools in a workshop (hammer, screwdriver) that allow you to modify materials. I/O is like the communication channels (speaking, listening) that allow you to interact with others.

3. Core Concept / Definition (The Truth)

Simple definition: **Operators** are symbols that tell the compiler to perform specific mathematical, relational, or logical operations and produce a result. **Input/Output (I/O)** refers to the communication between a computer system and the outside world, typically handled by functions like `printf()` for output and `scanf()` for input.

Key rule(s):

- Operators have precedence and associativity that determine the order of evaluation.
- `printf()` uses format specifiers to display different data types.

- `scanf()` requires the address of the variable (`&`) to store input.

4. Visual / Flow / Diagram (Show, Don't Tell)

Basic I/O Flow:

```
User -> Input Device (Keyboard) -> scanf() -> Program Variable
Program Variable -> printf() -> Output Device (Screen) -> User
```

Operator Precedence (Simplified):

```
1. Unary (++, --, !, -, &)
2. Arithmetic (*, /, %)
3. Arithmetic (+, -)
4. Relational (<, <=, >, >=)
5. Equality (==, !=)
6. Logical (&&)
7. Logical (||)
8. Assignment (=, +=, -=, etc.)
```

5. Syntax / Structure (The Tool)

Arithmetic Operators:

- `+` (Addition)
- `-` (Subtraction)
- `*` (Multiplication)
- `/` (Division)
- `%` (Modulo - remainder)

Relational Operators:

- `==` (Equal to)
- `!=` (Not equal to)

- `<` (Less than)
- `>` (Greater than)
- `<=` (Less than or equal to)
- `>=` (Greater than or equal to)

Logical Operators:

- `&&` (Logical AND)
- `||` (Logical OR)
- `!` (Logical NOT)

Assignment Operators:

- `=` (Simple assignment)
- `+=`, `-=`, `*=`, `/=`, `%=` (Compound assignment)

Increment/Decrement Operators:

- `++` (Increment by 1)
- `--` (Decrement by 1)

Input/Output Functions:

```c
// Output
printf("Format string with specifiers", var1, var2, ...);

// Input
scanf("Format string with specifiers", &var1, &var2, ...);
```

6. Example (The Action)

```c
#include <stdio.h>

int main() {
    int a = 10, b = 5;
    int sum, difference, product, quotient, remainder;
    int is_equal, is_greater;

    // Arithmetic Operations
    sum = a + b;         // sum = 15
    difference = a - b; // difference = 5
    product = a * b;     // product = 50
    quotient = a / b;    // quotient = 2
    remainder = a % b;  // remainder = 0

    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);

    // Relational and Logical Operations
    is_equal = (a == b);      // is_equal = 0 (false)
    is_greater = (a > b);     // is_greater = 1 (true)

    printf("Is a equal to b? %d\n", is_equal);
    printf("Is a greater than b? %d\n", is_greater);

    // Input Example
    int user_num;
    printf("Enter a number: ");
    scanf("%d", &user_num);
    printf("You entered: %d\n", user_num);

    return 0;
}
```

7. Common Mistakes (The Conflict)

1. **`=` vs `==`:** Using `=` (assignment) instead of `==` (equality comparison) in conditional statements, leading to logical errors.
2. **Missing `&` in `scanf()`:** Forgetting the address-of operator (`&`) before variable names in `scanf()`, causing runtime errors or crashes.
3. **Integer Division:** Expecting floating-point results from integer division (e.g., `5 / 2` results in `2`, not `2.5`). Cast to `float` or `double` for decimal results.
4. **Operator Precedence:** Not understanding operator precedence, leading to incorrect evaluation order (e.g., `a + b * c` evaluates `b * c` first).

8. Key Points / Rules (The Takeaway)

- C offers a rich set of operators for arithmetic, relational, logical, and assignment operations.
- `printf()` is used for formatted output to the console, while `scanf()` is used for formatted input from the console.
- Always use the correct format specifiers (`%d`, `%f`, `%c`, etc.) for I/O operations.
- Remember to use the `&` operator with variables in `scanf()`.
- Be mindful of operator precedence and associativity to ensure expressions are evaluated as intended.

9. One-Line Summary (The Ending)

Operators enable data manipulation, while I/O functions facilitate essential communication between the program and the user, making C programs dynamic and interactive.

10. (Optional) Practice Trigger

Question: Write a C program that calculates the area of a rectangle after taking its length and width as input from the user.

Variation Idea: Modify the program to also calculate the perimeter and display both results, ensuring proper formatting for floating-point numbers.