# Problem Solving With C

## SESSION 2: OPERATORS & INPUT/OUTPUT

### Chapter 1: Operators - Making Things Happen

#### What are Operators?

Operators are symbols that tell the computer to perform specific operations. Like mathematical symbols (+, -, ×, ÷), but more powerful!

#### Types of Operators in C

##### *1. Arithmetic Operators*

Perform mathematical calculations

| Operator | Name | Example | Result |
|---|---|---|---|
| + | Addition | 5 + 3 | 8 |
| - | Subtraction | 5 - 3 | 2 |
| * | Multiplication | 5 * 3 | 15 |
| / | Division | 10 / 3 | 3 (integer division) |
| % | Modulus (Remainder) | 10 % 3 | 1 |

**Important Division Rules:**

```
// Integer Division
int a = 10, b = 3;
int result = a / b;     // Result: 3 (decimal part lost)

// Float Division
float x = 10.0, y = 3.0;
float result2 = x / y;  // Result: 3.333...

// Mixed Division
int m = 10;
float n = 3.0;
float result3 = m / n;  // Result: 3.333...
```

**Common Question:** “What is modulus (%) used for?” **Answer:** - Check if number is even/odd: `num % 2 == 0` (even) - Get last digit: `num % 10` - Check divisibility: `num % 5 == 0` (divisible by 5)

## *2. Relational Operators*

Compare two values and return true (1) or false (0)

| Operator | Meaning | Example | Result |
|---|---|---|---|
| == | Equal to | 5 == 5 | 1 (true) |
| != | Not equal to | 5 != 3 | 1 (true) |
| > | Greater than | 5 > 3 | 1 (true) |
| < | Less than | 5 < 3 | 0 (false) |
| >= | Greater than or equal | 5 >= 5 | 1 (true) |
| <= | Less than or equal | 5 <= 3 | 0 (false) |

**Common Mistake:** Using = instead of ==

```
// WRONG
if (x = 5)     // This assigns 5 to x

// CORRECT
if (x == 5)    // This compares x with 5
```

## *3. Logical Operators*

Combine multiple conditions

| Operator | Name | Meaning | Example |
|---|---|---|---|
| && | AND | Both must be true | (5 > 3) && (2 < 4) = true |
| \|\| | OR | At least one true | (5 > 7) \|\| (2 < 4) = true |
| ! | NOT | Reverses true/false | !(5 > 3) = false |

**Truth Tables:**

AND (&&):

```
True  && True  = True
True  && False = False
False && True  = False
False && False = False
```

OR (||):

```
True  || True  = True
True  || False = True
False || True  = True
False || False = False
```

## *4. Bitwise Operators*

Operate on individual bits

| Operator | Name | Example |
|---|---|---|
| & | Bitwise AND | 5 & 3 = 1 |
| \| | Bitwise OR | 5 \| 3 = 7 |
| ^ | Bitwise XOR | 5 ^ 3 = 6 |
| ~ | Bitwise NOT | ~5 = -6 |
| << | Left shift | 5 << 1 = 10 |
| >> | Right shift | 5 >> 1 = 2 |

**Visual Example:**

```
5 in binary: 0101
3 in binary: 0011
-----------------
5 & 3:       0001 = 1
5 | 3:       0111 = 7
5 ^ 3:       0110 = 6
```

### *5. Assignment Operators*

Assign values to variables

| Operator | Example | Equivalent to |
|---|---|---|
| = | x = 5 | x = 5 |
| += | x += 3 | x = x + 3 |
| -= | x -= 3 | x = x - 3 |
| *= | x *= 3 | x = x * 3 |
| /= | x /= 3 | x = x / 3 |
| %= | x %= 3 | x = x % 3 |

### *6. Ternary Operator (? :)*

Shorthand for if-else

**Syntax:** `condition ? value_if_true : value_if_false`

```
int a = 20;
(a == 20) ? printf(“Yes”) : printf(“No”);
// If a == 20, print Yes, else print No

// Same as:
if (a == 20)
    printf(“Yes”);
else
    printf(“No”);
```

---

# Chapter 2: Input and Output Operations

## Output with printf()

### *Format Specifiers*

Tell printf() how to display different data types

| Specifier | Data Type | Example |
|---|---|---|
| %d | int | printf("%d", 10); |
| %f | float | printf("%f", 3.14); |
| %c | char | printf("%c", 'A'); |
| %s | string | printf("%s", "Hello"); |
| %lf | double | printf("%lf", 3.14159); |
| %x | hexadecimal | printf("%x", 255); |
| %o | octal | printf("%o", 8); |
| %% | print % | printf("%%"); |

## Input with scanf()

### *Basic Syntax*

```
scanf("format_specifier", &variable);
```

**Important:** Use & (address operator) before variable name!

### *Common Input Operations*

```c
#include <stdio.h>

int main()
{
    int age;
    float height;
    char grade;

    // Input integer
    printf("Enter your age: ");
    scanf("%d", &age);

    // Input float
    printf("Enter your height: ");
    scanf("%f", &height);

    // Input character
    printf("Enter your grade: ");
    scanf(" %c", &grade);  // Note: space before %c
    // Display
    printf("\nYour Details:\n");
    printf("Age: %d\n", age);
    printf("Height: %.1f\n", height);
    printf("Grade: %c\n", grade);

    return 0;
}
```

**Common Question:** “Why & before variable?” **Answer:** scanf() needs the memory address where to store the input. & gives the address of the variable.

**Common Question:** “Why space before %c in scanf?” **Answer:** To skip any leftover whitespace (like Enter key) from previous input.

*Multiple Inputs*

```
// Method 1: Separate scanf
int x, y;
scanf("%d", &x);
scanf("%d", &y);

// Method 2: Single scanf
scanf("%d %d", &x, &y);

// Method 3: With different types
int age;
float salary;
scanf("%d %f", &age, &salary);
```

### Common Input/Output Mistakes and Solutions

**Mistake 1:** Forgetting & in scanf

```
int num;
scanf("%d", num);    // WRONG! Will crash
scanf("%d", &num);   // CORRECT
```

**Mistake 2:** Wrong format specifier

```
float price;
scanf("%d", &price);  // WRONG! %d for int, not float
scanf("%f", &price);  // CORRECT
```

---

# Chapter 3: Frequently Asked Questions

### Q1: Why do we write return 0?

**Answer:** It tells the operating system that the program completed successfully. Non-zero values indicate errors.

### Q2: Can I use multiple data types in one printf?

**Answer:** Yes!

```
int age = 20;
float height = 5.8;
printf("Age: %d, Height: %.1f\n", age, height);
```

### Q3: What's the difference between 5 and '5'?

**Answer:** - `5` is an integer (numeric value) - `'5'` is a character (ASCII value 53)

### Q4: Why does 10/3 give 3 instead of 3.33?

**Answer:** Integer division truncates decimals. Use float:

```
float result = 10.0 / 3.0;  // Gives 3.333...
```

### Q5: Can variable names have spaces?

**Answer:** No. Use underscore: `student_age` or camelCase: `studentAge`

### Q6: What happens if I don't include stdio.h?

**Answer:** Compilation error - printf/scanf won't be recognized.

### Q7: Is C case-sensitive?

**Answer:** Yes. `age`, `Age`, and `AGE` are different variables.

### Q8: What's the difference between = and ==?

**Answer:** - `=` assigns value: `x = 5` - `==` compares values: `if (x == 5)`

### Q9: Why use float and double both?

**Answer:** - float: Less memory (4 bytes), sufficient for most cases - double: More precision (8 bytes), for scientific calculations

### Q10: Can I input multiple values in one line?

**Answer:** Yes:

```
scanf("%d %d %d", &a, &b, &c);
// User inputs: 10 20 30
```