Class Note: Operators (Session 5)

Overview

Operators are the building blocks of programming, enabling calculations, comparisons, and logical decisions. This session explores **arithmetic**, **relational**, **logical**, and **assignment** operators, showing how they are used in pseudocode and flowcharts to manipulate data and control program flow. By mastering operators, you will be able to create dynamic algorithms for real-world problems, building on the selection constructs from Session 4.

Learning Objectives

- Identify and apply arithmetic operators for mathematical computations.
- Use relational operators to compare values in decision-making.
- Combine conditions with logical operators to create complex logic.
- Apply assignment operators to update variable values.
- Write pseudocode and design flowcharts incorporating operators to solve practical problems.

Key Concepts

1. **What Are Operators?**
   - Operators are symbols or keywords that perform operations on operands (e.g., numbers, variables).
   - They are used in expressions to compute results, compare data, or evaluate conditions.
   - Example: In `area = length * width`, `*` is an arithmetic operator, and `=` is an assignment operator.
2. **Types of Operators**
   - **Arithmetic Operators**:
     - `+` (addition), `-` (subtraction), `*` (multiplication), `/` (division), `%` (modulo, remainder).
     - Example: `5 % 2` returns `1` (remainder of 5 ÷ 2).
   - **Relational Operators**:
     - `>` (greater than), `<` (less than), `>=` (greater than or equal to), `<=` (less than or equal to), `==` (equal to), `≠` (not equal to).
     - Example: `score >= 70` evaluates to true if `score` is 75.

- **Logical Operators**:
  - `AND` (both conditions true), `OR` (at least one condition true), `NOT` (negates a condition).
  - Example: `age >= 18 AND isCitizen == "Yes"` checks voting eligibility.
- **Assignment Operators**:
  - `=` (assign), `+=` (add and assign), `-=` (subtract and assign), `*=` (multiply and assign), `/=` (divide and assign).
  - Example: `counter += 1` increases `counter` by 1.

3. **Operator Precedence**
   - Operators are evaluated in this order:
     a. Parentheses `()`
     b. Arithmetic: `*`, `/`, `%` (left to right), then `+`, `-` (left to right)
     c. Relational: `>`, `<`, `>=`, `<=`, `==`, `≠`
     d. Logical: `NOT`, `AND`, `OR`
   - Example: `10 * 2 + 5` evaluates to `20 + 5 = 25` because `*` has higher precedence.
4. **Using Operators in Algorithms**
   - **Pseudocode**: Operators appear in assignments (e.g., `total = price * quantity`) and conditions (e.g., `IF score >= 60 THEN`).
   - **Flowcharts**: Arithmetic/assignment operations are in rectangles (process steps); relational/logical conditions are in diamonds (decision points).

Examples

1. **Arithmetic Operators: Calculate Circle Area**
   - **Problem**: Compute the area of a circle given its radius (area = π * radius², use π ≈ 3.14).
   - **Pseudocode**:

```
START
INPUT radius
SET pi = 3.14
area = pi * radius * radius
OUTPUT "Area of circle: ", area
END
```

   - **Flowchart** (text-based):

```
[Start] → [Input radius] → [pi = 3.14] → [area = pi * radius * radius] → [Output "Area of
```

   - **Explanation**: Uses `*` for multiplication to calculate the area.
2. **Relational Operators: Check Temperature Range**

- **Problem**: Determine if the temperature is comfortable (15°C ≤ temp ≤ 25°C).
- **Pseudocode**:

```
START
INPUT temperature
IF temperature >= 15 AND temperature <= 25 THEN
    OUTPUT "Comfortable temperature"
ELSE
    OUTPUT "Uncomfortable temperature"
END IF
END
```

- **Flowchart**:

```
[Start] → [Input temperature] → [temperature >= 15 AND temperature <= 25?] → Yes → [Outpu
                                                ↓ No
                                [Output "Uncomfortable temperature"] → [End]
```

- **Explanation**: Uses `>=`, `<=`, and `AND` to check the range.

3. **Logical Operators: Library Access**
   - **Problem**: Grant library access if the user is a member or has a guest pass.
   - **Pseudocode**:

```
START
INPUT isMember, hasGuestPass
IF isMember == "Yes" OR hasGuestPass == "Yes" THEN
    OUTPUT "Access granted"
ELSE
    OUTPUT "Access denied"
END IF
END
```

   - **Flowchart**:

```
[Start] → [Input isMember, hasGuestPass] → [isMember == "Yes" OR hasGuestPass == "Yes"?]
                                                ↓ No
                                [Output "Access denied"] → [End]
```

   - **Explanation**: `OR` allows access if either condition is true.

4. **Assignment Operators: Track Inventory**
   - **Problem**: Update inventory after adding new stock and selling items.

- **Pseudocode**:

```
START
INPUT currentStock, newStock, soldItems
SET inventory = currentStock
inventory += newStock
inventory -= soldItems
OUTPUT "Updated inventory: ", inventory
END
```

- **Flowchart**:

```
[Start] → [Input currentStock, newStock, soldItems] → [inventory = currentStock] → [inven
```

- **Explanation**: `+=` adds `newStock`, and `-=` subtracts `soldItems`.

Classwork Activities

1. **Pseudocode Writing**:
   - **Task**: Write pseudocode to calculate a restaurant bill:
     - Input: Meal cost, tip percentage (e.g., 15 for 15%).
     - Calculate: Tip = cost * (percentage / 100), total = cost + tip.
     - Output: Total bill.
   - **Expected Pseudocode**:

```
START
INPUT mealCost, tipPercentage
tip = mealCost * (tipPercentage / 100)
total = mealCost + tip
OUTPUT "Total bill: ", total
END
```

2. **Flowchart Design**:
   - **Task**: Create a flowchart to check if a number is divisible by 3 and 5.
     - Use modulo (`%`) and logical operators (`AND`).
     - Output "Divisible by both", "Divisible by 3 only", "Divisible by 5 only", or "Not divisible".
   - **Expected Flowchart** (text-based):

```
[Start] → [Input number] → [number % 3 == 0 AND number % 5 == 0?] → Yes → [Output "Divisi
                                        ↓ No
                                   [number % 3 == 0?] → Yes → [Output "Divisible by 3 only
                                        ↓ No
                                   [number % 5 == 0?] → Yes → [Output "Divisible by 5 only
                                        ↓ No
                                   [Output "Not divisible"] → [End]
```

3. **Error Correction**:
   - **Task**: Fix errors in the following pseudocode:

```
INPUT hoursWorked
IF hoursWorked > 40 THEN
    overtimePay = (hoursWorked - 40) * 15
    totalPay = (40 * 10) + overtimePay
ELSE
    totalPay = hoursWorked * 10
OUTPUT totalPay
END
```

   - **Issues**:
     - Missing `END IF`.
     - `OUTPUT` is not properly aligned (should be inside `ELSE` block).
     - No `START/END`.
   - **Corrected Pseudocode**:

```
START
INPUT hoursWorked
IF hoursWorked > 40 THEN
    overtimePay = (hoursWorked - 40) * 15
    totalPay = (40 * 10) + overtimePay
ELSE
    totalPay = hoursWorked * 10
END IF
OUTPUT "Total pay: ", totalPay
END
```

4. **Group Activity: Real-World Scenario**:
   - **Task**: In pairs, design pseudocode and a flowchart for a program that determines a gym membership fee:
     - $30/month: Age ≥ 18 and not a senior (age < 65).

    - $20/month: Senior (age ≥ 65) or student (age 16–22).
    - $0/month: Under 16 (free junior membership).
    - Use relational (`>=` , `<`) and logical (`AND` , `OR`) operators.
- **Example Pseudocode**:

```
START
INPUT age
IF age < 16 THEN
    OUTPUT "Free junior membership"
ELSE IF (age >= 16 AND age <= 22) OR age >= 65 THEN
    OUTPUT "Membership fee: $20"
ELSE
    OUTPUT "Membership fee: $30"
END IF
END
```

- **Discussion**: Share solutions and discuss how operator precedence affects conditions (e.g., parentheses in `(age >= 16 AND age <= 22)`).

Objective Questions (Multiple Choice)

Test your understanding of operators with the following questions:

1. What does the `/` operator do?
    - A) Returns the remainder
    - B) Divides two numbers
    - C) Adds two numbers
    - D) Compares equality
    - **Answer**: B
2. Which operator checks if a value is less than or equal to another?
    - A) >
    - B) <=
    - C) ==
    - D) ≠
    - **Answer**: B
3. What is the result of `3 + 4 * 2 - 1` ?
    - A) 10
    - B) 14
    - C) 9

- D) 12
- **Answer**: C (Explanation: `*` first: `4 * 2 = 8`, then `3 + 8 - 1 = 11 - 1 = 10`)

4. What does this pseudocode output if score = 85, isExtraCredit = "Yes"?

```
IF score >= 80 OR isExtraCredit == "Yes" THEN
    OUTPUT "Good job"
ELSE
    OUTPUT "Try harder"
END IF
```

- A) Good job
- B) Try harder
- C) No output
- D) Error
- **Answer**: A

5. What does the `*=` operator do?
   - A) Multiplies a variable by a value and assigns the result
   - B) Checks if two values are equal
   - C) Divides a variable by a value
   - D) Negates a condition
   - **Answer**: A
6. Which logical operator negates a condition?
   - A) AND
   - B) OR
   - C) NOT
   - D) NONE
   - **Answer**: C
7. What is wrong with this pseudocode?

```
INPUT distance
IF distance < 5 THEN
    cost = distance * 2
ELSE
    cost = distance + 5
OUTPUT "Cost: ", cost
END
```

- A) Incorrect operator
- B) Missing END IF
- C) No condition in ELSE

- D) No arithmetic operator
- **Answer**: B

8. In a flowchart, where are relational operators used?
   - A) Rectangle
   - B) Oval
   - C) Diamond
   - D) Parallelogram
   - **Answer**: C
9. What is the output of this pseudocode if quantity = 10?

```
quantity *= 2
OUTPUT quantity
```

   - A) 10
   - B) 20
   - C) 5
   - D) 0
   - **Answer**: B
10. What does this pseudocode do if x = 7, y = 3?

```
IF x % y == 1 THEN
    OUTPUT "Remainder is 1"
ELSE
    OUTPUT "Remainder is not 1"
END IF
```

    - A) Remainder is 1
    - B) Remainder is not 1
    - C) No output
    - D) Error
    - **Answer**: A (Explanation: `7 % 3 = 1`, so condition is true)

Homework

1. **Pseudocode Practice**:
   - Write pseudocode to calculate a simple interest:
     - Input: Principal, rate (%), time (years).
     - Calculate: Interest = (principal * rate * time) / 100.
     - Output: Interest.

2. **Flowchart Creation**:
   - Design a flowchart to check if a number is even and greater than 10.
     - Use modulo (`%`) and relational (`>`) operators.
     - Output "Valid number" or "Invalid number".
3. **Real-World Application**:
   - Describe a scenario where operators are used (e.g., calculating utility bills). Write a short paragraph and pseudocode for it.
4. **Debugging Challenge**:
   - Fix this pseudocode:

```
INPUT price, quantity
total = price * quantity
IF total > 100 THEN
    discount = total * 0.1
    total =- discount
OUTPUT total
END
```

     - Issues: Incorrect operator (`=-` instead of `-=`), missing `END IF` , no `ELSE` , no `START/END` .
     - Provide the corrected version.

Additional Notes

- **Teaching Tips**:
  - **Duration**: 2–3 class periods (3–4.5 hours).
    - **Period 1**: Arithmetic and assignment operators, Example 1 and 4, Activity 1.
    - **Period 2**: Relational and logical operators, Examples 2 and 3, Activities 2 and 3.
    - **Period 3 (optional)**: Group Activity 4, review, quiz.
  - **Visual Aids**: Demonstrate precedence with examples (e.g., `2 + 3 * 4`) on a whiteboard.
  - **Engagement**: Use relatable scenarios (e.g., restaurant bills, gym memberships) to make operators relevant.
  - **Differentiation**:
    - Beginners: Focus on single operators (e.g., `+` , `>`) before combining them.
    - Advanced: Challenge with complex conditions (e.g., `IF (age >= 18 AND isMember == "Yes") OR hasGuestPass == "Yes" THEN`).
- **Resources**:
  - Flowchart tools: Draw.io, Lucidchart, or paper templates.
  - Text editor for pseudocode: Notepad++, Visual Studio Code.
- **Assessment**:

    - Use objective questions for a quiz or homework.
    - Grade classwork for correct operator use and logical flow.
- **Extension**:
    - Implement a pseudocode example in Python:

```
radius = float(input("Enter radius: "))
pi = 3.14
area = pi * radius * radius
print("Area of circle:", area)
```

- **Connections**:
    - Builds on Session 4 (Selection Constructs) for relational/logical operators.
    - Prepares for Session 6 (Iteration Constructs) by introducing expressions for loop conditions.