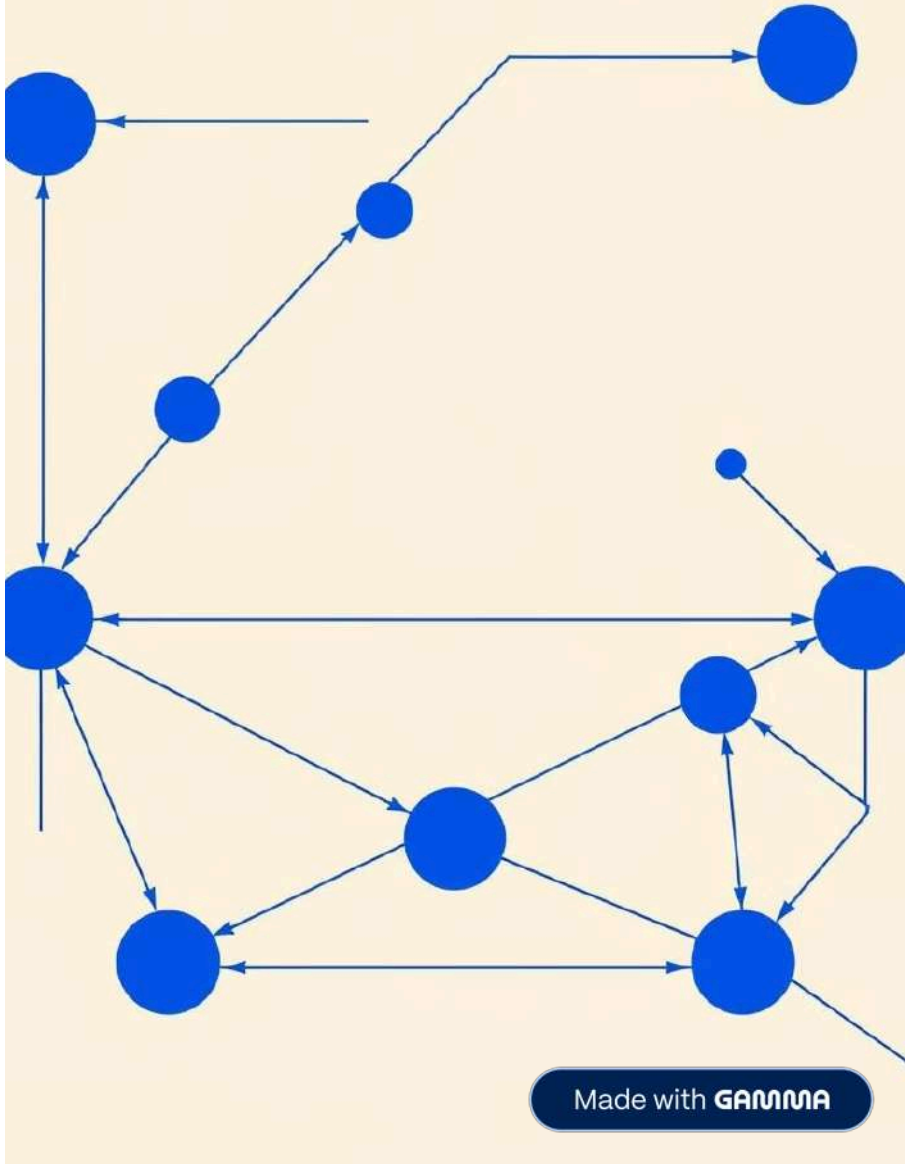
Lecture 3: Control Structures

Date: Week 3 | Duration: 1 hour



Lecture Objectives

1

Execution Flow

Understanding the concept of program execution flow control.

2

Flowcharts

Ability to create and read flowcharts for branching algorithms.

3

Loops

Knowledge of types of cyclic constructs and their application.

4

Nested Structures

Understanding the principles of building nested structures.

5

Construct Selection

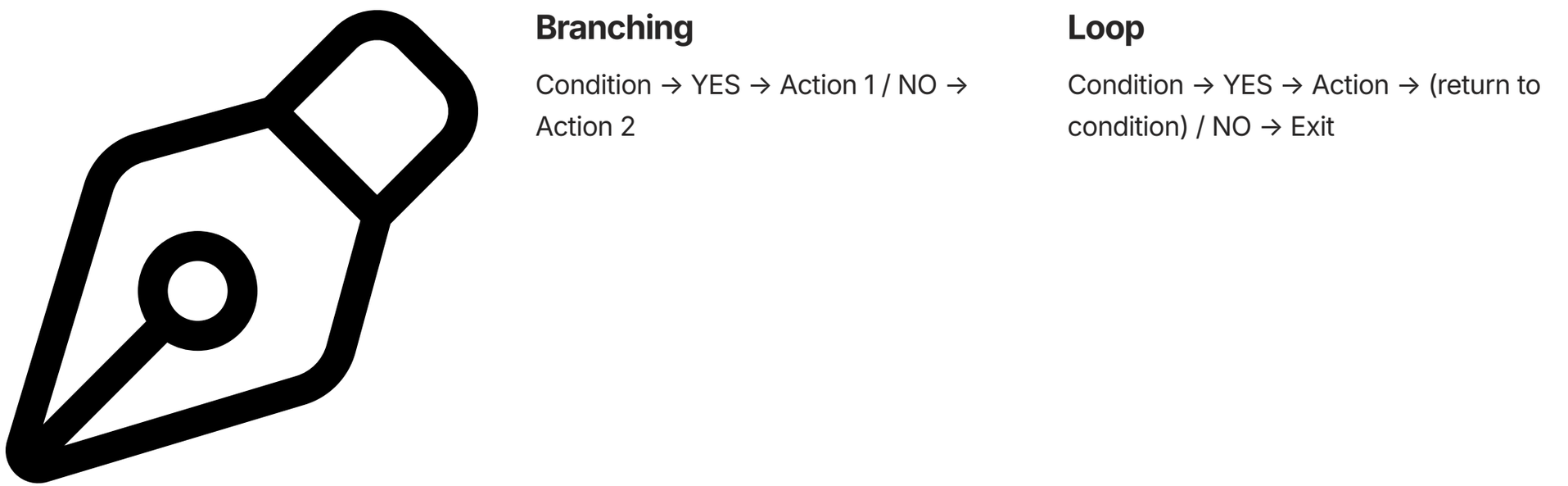
Ability to choose the appropriate control construct for solving problems.

1. Control Flow

What is Control Flow?

Control Flow is the order in which a program's instructions are executed.

Types of Control Structures



Sequence

Instruction 1 → Instruction 2 → Instruction 3

Branching

Condition → YES → Action 1 / NO → Action 2

Loop

Condition → YES → Action → (return to condition) / NO → Exit

Structured Programming Theorem

Böhm-Jacopini theorem (1966): Any algorithm can be implemented using only three control structures:

- Sequence
- Selection (Branching)
- Iteration (Loop)

This is the foundation of structured programming — a methodology for writing clear and reliable programs.

Control Flow Block Diagrams

A flowchart is a graphical representation of an algorithm, showing the flow of control.

2. Branching Algorithms

Concept of Branching

> Branching is a control structure that allows choosing one of alternative execution paths based on a condition.

Types of Branching

Incomplete Branching (if)

IF condition THEN action END_IF

```
IF number < 0 THEN
  number = -number
END_IF
```

Complete Branching (if-else)

IF condition THEN action1 ELSE action2 END_IF

```
IF number MOD 2 = 0 THEN
 PRINT "Even"
ELSE
 PRINT "Odd"
END_IF
```

3. Looping Algorithms

Concept of a Loop

A loop is a control structure that allows a block of instructions to be executed repeatedly.

Components of a loop:

- Initialization: initial setup of variables
- Condition: check for continuation/termination
- Loop body: actions to be repeated
- Modification: changing the control variable

Types of Loops

Pre-condition loop (WHILE)

The condition is checked before the body is executed. May not execute even once.

```
WHILE condition DO action
END_WHILE
```

Post-condition loop (DO-WHILE)

The condition is checked after the body is executed. Will execute at least once.

```
DO action WHILE condition
```

Parameter loop (FOR)

The number of iterations is known. Automatic counter management.

```
FOR i FROM start TO end STEP
step DO action END_FOR
```

Loop Control and Nested Structures

Early Exit (break)

Immediately terminates loop execution.

```
WHILE true
  IF x = 0 THEN BREAK
END_WHILE
```

Skip to Next Iteration (continue)

Skips the rest of the loop body and proceeds to the next iteration.

```
FOR i FROM 1 TO 10
  IF i MOD 2 = 0 THEN CONTINUE
END_FOR
```

Nested Conditions

Conditions located within other conditions.

```
IF condition1 THEN
  IF condition2 THEN
    action1
  ELSE
    action2
END_IF
```

Nested Loops

Loops located within other loops.

```
FOR i FROM 1 TO 10
 FOR j FROM 1 TO 10
 product = i * j
END_FOR
```

Infinite Loops

An infinite loop is a programming construct where a block of code repeats indefinitely because the loop's exit condition never becomes false. They can be either intentional or erroneous.

Intentional Infinite Loops

Used in systems that must run continuously, such as operating systems, servers, embedded systems, or GUI applications waiting for user interaction.

```
// Server waiting for requests
WHILE true
    request = WAIT_FOR_REQUEST()
    PROCESS(request)
END_WHILE
```

Accidental Infinite Loops (Errors)

Occur due to errors in program logic when the loop termination condition is never met. This leads to program freezing or memory overflow.

```
// ERROR: i does not change
i = 1
WHILE i <= 10
    PRINT i
    // Forgot i = i + 1
END_WHILE
```

4. Nested Structures

Nested structures are control constructs (e.g., conditional statements or loops) that are placed inside other similar or different control constructs. This allows for the creation of more complex and detailed algorithms for data processing, increasing program flexibility and power.

Nested Conditional Statements

Nested conditional statements are used when it is necessary to check several conditions sequentially, where the result of one check affects the possibility or method of executing the next. This allows for handling various scenarios depending on multiple factors.

```
// Determine the quarter of the year based on the month number
IF month >= 1 AND month <= 12 THEN
   IF month >= 1 AND month <= 3 THEN
      OUTPUT "Month belongs to the first quarter."
   ELSE IF month >= 4 AND month <= 6 THEN
      OUTPUT "Month belongs to the second quarter."
   ELSE IF month >= 7 AND month <= 9 THEN
      OUTPUT "Month belongs to the third quarter."
   ELSE // month >= 10 AND month <= 12
      OUTPUT "Month belongs to the fourth quarter."
   END IF
ELSE
   OUTPUT "Invalid month number entered. The month number must be from 1 to 12."
END IF
```

Nested Loops

Nested loops are loops that are located inside other loops. The inner loop fully executes for each iteration of the outer loop. They are often used for working with two-dimensional data structures, such as matrices, or for creating combinatorial sequences.

```
// Output a 10x10 multiplication table
FOR i FROM 1 TO 10 LOOP
   FOR j FROM 1 TO 10 LOOP
      OUTPUT i * j, " " // Output product and a space
   END LOOP
   OUTPUT " " // New line after each row of the table
END LOOP
```

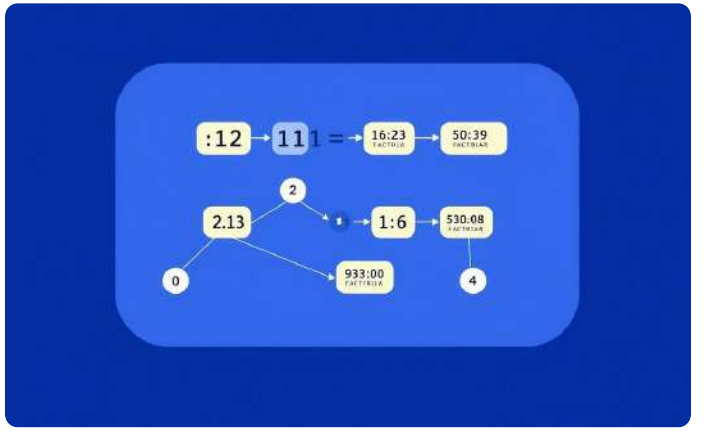
5. Practical Algorithm Examples

Finding Prime Numbers

An algorithm for finding all prime numbers up to a given N.

Palindrome Check

Determines if a string is a palindrome (reads the same forwards and backwards).



Factorial Calculation

An iterative approach to calculate the factorial of a number N.

1. Prime Number Search Algorithm (Sieve of Eratosthenes)

This algorithm efficiently finds all prime numbers up to a given value `n`, using the "sieve" method.

```
START
  INPUT n
  FOR number FROM 2 TO n
    is_prime = true
    FOR divisor FROM 2 TO sqrt(number)
      IF number MOD divisor = 0 THEN
        is_prime = false
        BREAK
      END_IF
    END_FOR
    IF is_prime THEN
      PRINT number
    END_IF
  END_FOR
END
```

2. Palindrome Checker Algorithm

A palindrome is a word, phrase, or sequence of characters that reads the same forwards and backward. This algorithm efficiently determines if a given string is a palindrome by comparing characters from both ends.

```
START
    INPUT string
    length = LENGTH(string)
    is_palindrome = true

    FOR i FROM 0 TO length/2
        IF string[i] ≠ string[length-1-i] THEN
            is_palindrome = false
            BREAK
        END_IF
    END_FOR

    IF is_palindrome THEN
        OUTPUT "Palindrome"
    ELSE
        OUTPUT "Not a palindrome"
    END_IF
END
```

3. Factorial Calculation Algorithm

The factorial of a number `n` (denoted as `n!`) is the product of all positive integers from 1 to `n`. An iterative approach to calculating the factorial involves multiplying successive numbers, starting from 1, until reaching the given `n`.

```
START
  INPUT n
  IF n < 0 THEN
    OUTPUT "Error: negative number. Factorial is defined only for non-negative numbers."
  ELSE IF n = 0 THEN
    OUTPUT 1 // Factorial of 0 is 1 by definition
  ELSE
    factorial = 1
    FOR i FROM 1 TO n
      factorial = factorial * i
    END_FOR
    OUTPUT factorial
  END_IF
END
```

Key Takeaways

1 **Changing Order**

Control structures alter the sequential order of execution.

2 **Three Basic Constructs**

Sequence, branching, loop — sufficient for any algorithm.

3 **Branching**

Allows choosing the program execution path.

4 **Loops**

Automate repetitive actions.

5 **Nesting**

Allows for creating complex algorithms.

6 **Flowcharts**

Help visualize program logic.

7 **Choosing the Construct**

The correct choice of construct simplifies problem-solving.

Review Questions and Resources

Review Questions

- What are the three basic control structures?
- What is the difference between a pre-test loop and a post-test loop?
- What is an infinite loop and when is it needed?
- How do nested loops work?
- Which loop should you choose if the number of iterations is known in advance?

Next Lecture: Arrays

Self-study Materials

- Lafore R. "Object-Oriented Programming in C++" Chapter 3-4
- Cormen T. "Introduction to Algorithms" Chapter 2.1-2.2
- Algorithm visualization: **https://visualgo.net**
- Flowchart builder: **https://www.draw.io**