Sequence Control

Any algorithm/program can be clearer and easy to understand kapag gumamit tayo ng self-contained modules called logic/control structures.

Control Structures

- provide the framework within which operations and data are combined into programs and sets of programs.
- control of the order of execution of the operations, both primitive and user defined, which we term sequence control and control of the transmission of data among the subprograms of a program is called **data control.**

Sequence Control

- the control of the order of execution of the operation
- Example:

```java
public class SequenceControlExample {
    public static void main(String[] args) {
        int x = 10;
        if (x > 5) {
            System.out.println("x is greater than 5");
        } else {
            System.out.println("x is not greater than 5");
        }
        System.out.println("End of program");
    }
}
```

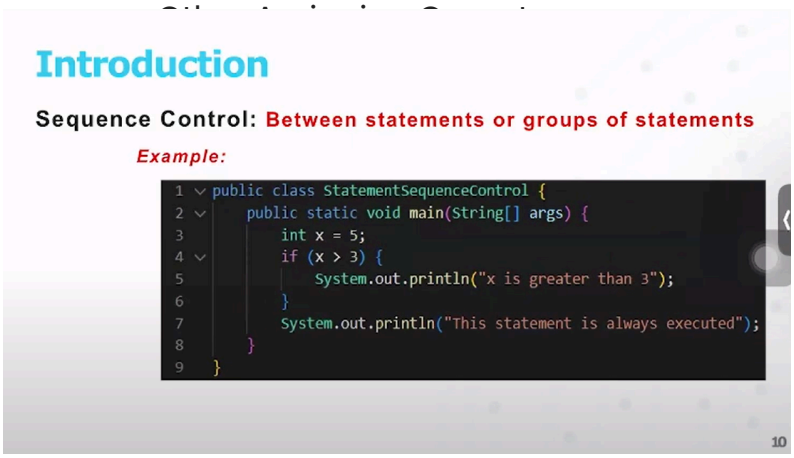
- determines the order of tasks to execute within the program. Defines line by line implementation by which statements are implemented **sequentially.**
- The use of **sequence control** structure can be categorized into **(3) three**. These are structures to be used in:
    - **Expressions**
        - how the data are manipulated using the precedence rules.

```java
public class ExpressionSequenceControl {
    public static void main(String[] args) {
        int result = 10 * 5 + 8;
        System.out.println("Result: " + result);
    }
}
```

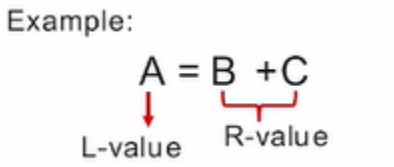
Sequence of operations within the expression.

- **Between statements or groups of statements**
  - **Basic Statements**
    - Statements that apply operations to data objects.
    - Considered as a unit of step

      Examples:
      - Assignment to Data Objects
      - Input / Output Statement



  - **Assignment to Data Object**
    - We use assignment statement to assign to the L-value (memory location) the R-value (the data object value) of some expression
    - Assignment statement is the central operation defined for every elementary data type. Its syntax varies in different programming languages

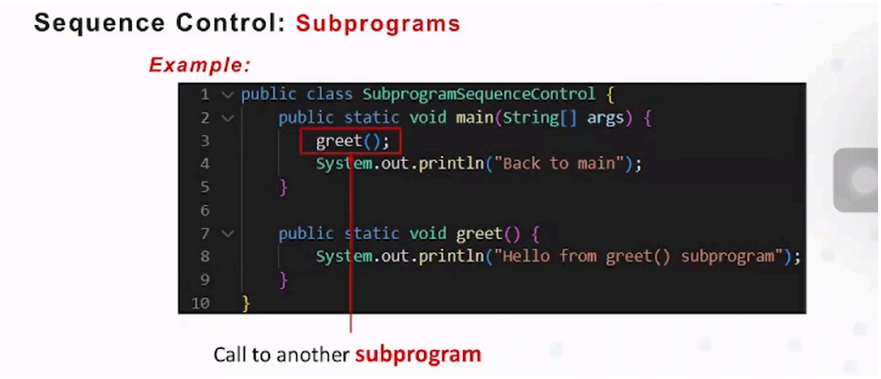


  - **Input and Output Statements**
    - **Input statement** - reads the data from the CRT, files, or communication lines. This statement can change the value of variables through assignments.
    - **Output Statement** - produces some form of output to let the user know what the “answer” is. In True BASIC the primary mechanism

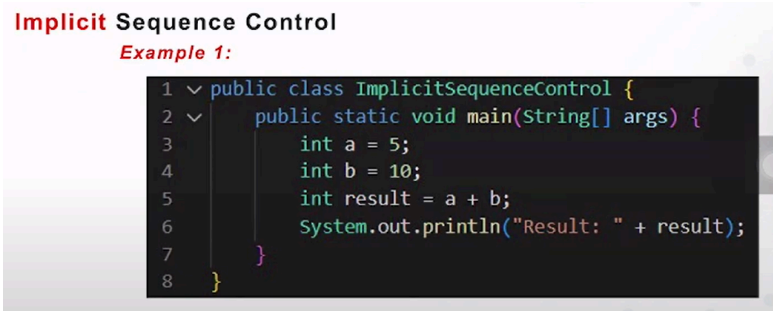
for generating output is the PRINT statement in some programming languages.

- **Other Assigning Operations**
  - Operations like *parameter transmission,* an assignment operation that assigns an argument value to the formal parameter
  - **Parameter transmission** is often defined as assignment of the argument value to the formal parameter.

- **Between subprograms**

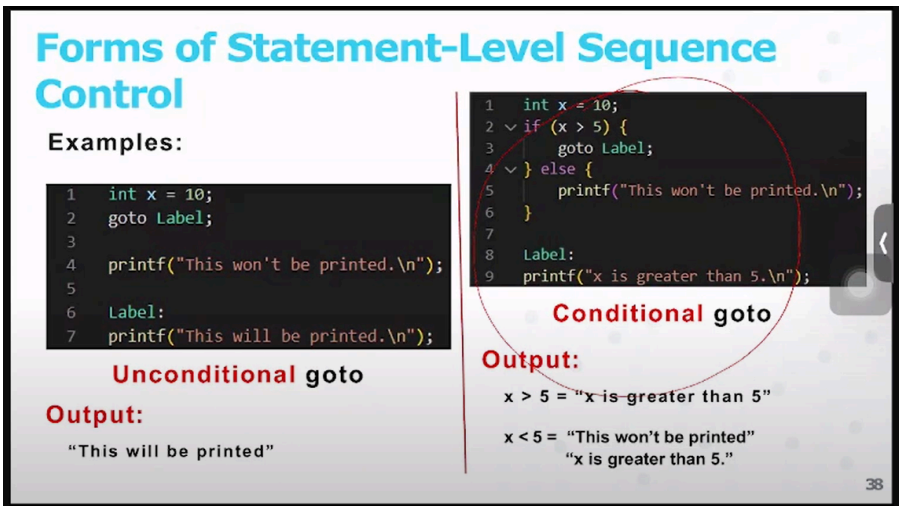


- Types of Sequence Control:
  - **Implicit Sequence Control Structure** - sequence structure defined by the language

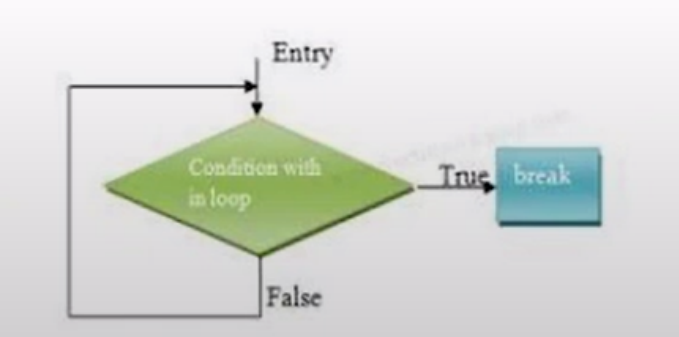


  - **Explicit Sequence Control Structure** - the programmer may optionally modify the implicit sequence or operation defined by the language.
    - **Goto Statement**
      - **Unconditional goto**
        - transfers control to another statement
        - All programs have started at the **top** and worked down to the bottom.

- **Conditional goto** - transfers control to another statement if a condition is satisfied.



- **Break Statement**
  - It causes control to move forward in the program to an explicit point at the end of a given control structure.



- **Compound Statement**
  - A sequence of statements that can be treated as a single statement.
  - Can be written within the compound statement.
  - Implemented in a computer by placing blocks of executable code representing each constituent statement in sequence memory.

Sequencing with Arithmetic Expression

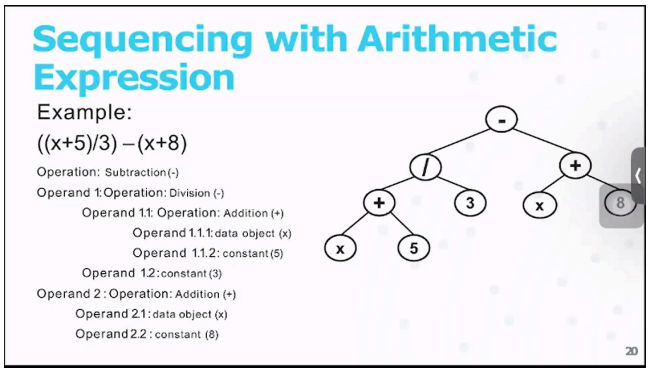
Example#1

A = (B + D) / K (this can be translated to a computer in different ways depending on the programming language that you are going to use.)

- **Functional Composition** - Basic sequence-control mechanism.

  Given an operation with its operands, the operands may be:

  - Constants
  - Data Objects
  - Other Operations



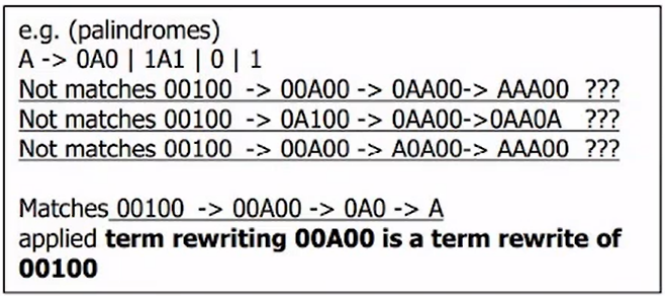
The representation of operations clarifies the control structure of the expression.

Linear presentation of the expression tree:

- **Prefix Notation**
  - Also known as **"Polish Notation".**
  - The operation symbol is ***first written*** then the operands in order from left to right follows.
- **Postfix Notation**
  - This is a suffix or **reverse polish**.
  - The ***reverse*** of the prefix, the operands will be written first then the operation symbol.
- **Infix / Suffix Notation**
  - Commonly used for binary operation
  - In their notation, the operation symbol is written between the operands.

Sequencing with non-Arithmetic Expression

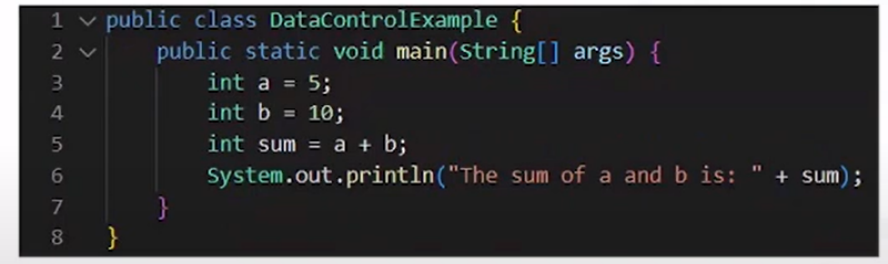
- Although every programming language uses arithmetic expressions, we have to consider that some languages, like Prolog, use other expression formats.
- **Prolog** is designed for processing character data, including other forms of expression evaluation.
- **Pattern Matching**
  - An operation succeeds by matching and assigning a set of variables to a predefined template.



- **Unification**
  - Prolog consists of facts, rules, and query (Unification).
  - The substitution of variables in relation to pattern match in order to determine if the query has a valid substitution consistent with the rules and facts in the database.
- **Backtracking**
  - An algorithm that tries all possible match searching for success, by successively ***stacking*** and ***unstacking*** partial results.
  - This can be unified with a rule in the database, where unification rules facts.

Data Control

- controls the transmission of data among the subprograms of a program.
- **Subprograms**: functions, variable assignments, methods, modules & packages.
- Example:



```java
public class DataControlExample {
    public static void main(String[] args) {
        int a = 5;
        int b = 10;
        int sum = a + b;
        System.out.println("The sum of a and b is: " + sum);
    }
}
```