Pseudo Code

What is Pseudocode

- Pseudocode is an informal high-level description of a computer program or algorithm.
- A Pseudocode is defined as a step-by-step description of an algorithm.
- Pseudocode does not use any programming language in its representation. It uses simple English language text as it is intended for human understanding rather than machine reading.
- Pseudocode is the intermediate state between an idea and its implementation(code) in a high-level language.

Why use pseudo code?

- **It's easier to read.** provides an easy way for people to understand the basics of how software functions without having to learn computer programming.
- **It simplifies code construction.** When the programmer goes through the process of developing and generating pseudocode converting that into real code written in any programming language will become much easier and faster.
- **It's a good middle point between idea and code.** Moving directly from the idea to the code is not always a smooth ride. That's where pseudocode presents a way to make the transition between the different stages somewhat simpler.
- **It's an important part of a document.** Pseudocode can be used in many scenarios to describe an algorithm clearly, such as in reports or papers.
- **It allows for quick bug detection.** Since pseudocode is written in a human-readable format, it is easier to edit and discover bugs before actually writing a single line of code.

Note

- Although pseudocode is a syntax-free description of an algorithm, it must provide a full description of the algorithm's logic so that moving from pseudocode to implementation is merely a task of translating each line into code using the syntax of any given programming language.

How

- Always capitalize the initial word (often one of the main six constructs).
- Make only one statement per line.
- Indent to show hierarchy, improve readability, and show nested constructs.
- Always end multi-line sections using any of the END keywords (ENDIF, ENDWHILE, etc.).
- Keep your statements' programming language independent.
- Use the naming domain of the problem, not that of the implementation. For instance: "Append the last name to the first name" instead of "name = first+ last."
- Keep it simple, concise, and readable.

Mathematical operations

- **Assignment:** ←

Example: $c \leftarrow 2\pi r$,

- **Comparison:** =, ≠, <, >, ≤, ≥
- **Arithmetic:** +, −, ×, /, mod
- **Floor/ceiling:** ⌊, ⌋, ⌈, ⌉

$a \leftarrow \lfloor b \rfloor + \lceil c \rceil$

- **Logical:** and, or
- **Sums, products:** Σ Π

Example: $h \leftarrow \Sigma a \in A\ 1/a$

Keywords

In Pseudocode, keywords are used to indicate common input-output and processing operations. They are written fully in uppercase.

- **START:** This is the start of your pseudocode.
- **INPUT:** This is data retrieved from the user through typing or through an input device.
- **READ / GET:** This is input used when reading data from a data file.
- **PRINT, DISPLAY, SHOW:** This will show your output to a screen or the relevant output device.
- **COMPUTE, CALCULATE, DETERMINE:** This is used to calculate the result of an expression.
- **SET, INIT:** To initialize values
- **INCREMENT, BUMP:** To increase the value of a variable
- **DECREMENT:** To reduce the value of a variable

Main constructs of pseudocode

- **SEQUENCE:** represents linear tasks sequentially performed one after the other.
- **WHILE:** a loop with a condition at its beginning.
- **REPEAT-UNTIL:** a loop with a condition at the bottom.
- **FOR:** another way of looping.
- **IF-THEN-ELSE:** a conditional statement changing the flow of the algorithm.
- **CASE:** the generalization form of IF-THEN-ELSE.

PSEUDOCODE CONSTRUCTS

SEQUENCE

Input: READ, OBTAIN, GET
Output: PRINT, DISPLAY, SHOW
Compute: COMPUTE, CALCULATE, DETERMINE
Initialize: SET, INIT
Add: INCREMENT, BUMP
Sub: DECREMENT

FOR

```
FOR iteration bounds
sequence
ENDFOR
```

WHILE

```
WHILE condition
sequence
ENDWHILE
```

CASE

```
CASE expression OF
condition 1: sequence 1
condition 2: sequence 2
...
condition n: sequence n
OTHERS:
default sequence
ENDCASE
```

REPEAT-UNTIL

```
REPEAT
sequence
UNTIL condition
```

IF-THEN-ELSE

```
IF condition THEN
    sequence 1
ELSE
    sequence 2
ENDIF
```

https://builtin.com/sites/www.builtin.com/files/styles/ckeditor_optimize/public/inline-images/national/pseudocode%2520constructs.png

PSEUDOCODE EXAMPLE

NOT BAD

```
FOR X = 1 to 10
  FOR Y = 1 to 10
    IF gameBoard[X][Y] = 0
      Do nothing
    ELSE
      CALL theCall(X, Y) (recursively)
      counter += 1
    END IF
  END FOR
END FOR
```

GOOD

```
SET moveCount to 1
FOR each row on the board
  FOR each column on the board
    IF gameBoard position (row, column) is occupied THEN
      CALL findAdjacentTiles with row, column
      INCREMENT moveCount
    END IF
  END FOR
END FOR
```

https://builtin.com/sites/www.builtin.com/files/styles/ckeditor_optimize/public/inline-images/national/pseudocode%2520example_0.png

Which is not good?

```
For each index of string
    IF string(index) is equal to goal
        RETURN "It found"
    END IF
END FOR
```

```
For i=0 to end
    IF string(i) = goal
        RETURN "It found"
    END IF
END FOR
```

```
For each index of string
    if string(index) = goal
        RETURN "It found"
    END if
END FOR
```

```
For each index of string
    IF string(index) is equal to goal
        Get the answer
    END IF
END FOR
```

Which is not good?

```
For each index of string
    IF string(index) is equal to goal
        RETURN "It found"
    END IF
END FOR
```

```
For i=0 to 20
    IF string(i) = "#"
        RETURN "It found"
    END IF
END FOR
```

```
For each index of string
    if string(index) = goal
        RETURN "it found"
    END if
END FOR
```

```
For each index of string
    IF string(index) is equal to goal
        Get the answer
    END iF
END FOR
```