Pseudocodes

CS115

Problem Solving Stages

Goal: Go from an idea on how to solve the problem to a valid and working implementation of that idea.

- Design/Validate an algorithm
- Apply the algorithm to the problem at hand
- Test it for various input

Pseudocodes

While designing/validating an algorithm it is important to focus on the thought process behind the algorithm, thinking about way on how it will or won't work instead of focusing on how to correct our syntax.

Pseudocode is a technique used to describe the distinct steps of an algorithm in a manner that is easy to understand from anyone with basic programming knowledge.

Why Pseudocodes?

- Better readability
- Ease up code construction
- Act as a start point for documentation
- Easier bug detection and fixing

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
```

- **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.

Rules for writing pseudocode

- Always **capitalize** the initial word (often one of the main 6 constructs).
- Have only **one** statement per line.
- **Indent** to show hierarchy, improve readability, and show nested construct
- Always **end** multiline 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**. E.g., "*Append the last name to the first name*" instead of "*name = first+ last.*"
- Keep it **simple**, **concise**, and **readable**.

Credits: Sara A. Metwalli

https://towardsdatascience.com/pseudocode-101-an-introduction-to-writing-good-pseudocode-1331cb855be7