CISP 300: Suggested Pseudocoding Keywords for Algorithms

CONSTANT Define a symbolic (named) constant. Values for constants must be defined in the definition. You may not use SET to define a value for a constant.

```
CONSTANT Real Acceleration = 32.3
CONSTANT Integer MAX_SCORES = 10
```

DECLARE Define a variable to contain data.

```
DECLARE Real price   // Simple variables
DECLARE Integer quantity
DECLARE String name
DECLARE Real testScores[MAX_SCORES]  // Declares an array
```

PRINT Send output to a printer.

DISPLAY Send output to a computer screen.

PROMPT Alternate form of `DISPLAY` used for brief messages to prompt for operator input from the keyboard device.

INPUT Receive input from the keyboard.

```
PROMPT "Please enter your name: "
INPUT name
```

SET Store a value to a previously declared variable.

```
SET X ← 52 + 8 - Y
SET X = 52 + 8 - Y
```

OPEN ... READ/WRITE ... CLOSE

Open a file for reading:

```
DECLARE InputFile myFile
DECLARE Integer num
OPEN myFile "inventory.dat"
READ myFile num
CLOSE myFile
```

Opening for writing:

```
DECLARE OutputFile salesFile
OPEN salesFile "sales.dat"
WRITE salesFile 32430.34
CLOSE salesFile
```

Define a module:

```
MODULE sum(Integer num1, Integer
   num2, Integer Ref result)
    SET result = num1 + num2
END MODULE
```

Call, or execute a module:

```
DECLARE Integer answer
CALL showMessage(3, 4, answer)
```

Define a function:

```
FUNCTION Integer sum(Integer num1,
  Integer num2)
    DECLARE Integer result
    SET result = num1 + num2
    RETURN result
END FUNCTION
```

Call, or execute a function:

```
DECLARE Integer answer
answer = sum(3, 4)
```

Single alternative Selection

```
IF ... THEN
    ***block of statements***
ENDIF
```

Dual alternative Selection

```
IF ... THEN
    ***block of statements***
ELSE
    ***block of statements***
ENDIF
```

CISP 300: Suggested Pseudocoding Keywords for Algorithms

WHILE *condition* ... END WHILE

Repetition. Test is performed *before* each iteration of loop. Indent the sequence of statements surrounded by the WHILE and END WHILE clauses:

```
WHILE condition
    ***block of statements***
END WHILE
```

DO ... WHILE *condition*

Repetition. Test is performed *after* each iteration of loop. Indent the sequence of statements surrounded by the DO and WHILE clauses:

```
DO
    ***block of statements***
WHILE condition
```

REPEAT ... UNTIL *condition*

Repetition. Test is performed *after* each iteration of loop. Indent the sequence of statements surrounded by the REPEAT and UNTIL clauses:

```
REPEAT
    ***block of statements***
UNTIL condition
```

FOR ... END FOR

Repetition. Test is performed *before* each iteration of loop. Indent the sequence of statements surrounded by the FOR and END FOR:

```
FOR counterVariable = startingValue TO maxValue
    ***block of statements***
END FOR condition
```

You may also specify an explicit increment. The increment can be positive or negative.

```
FOR counterVariable = startingValue TO maxValue STEP 5
    ***block of statements***
END FOR condition
```

< Boolean test to see if the expression on the left is less than the expression on the right.

> Boolean test to see if the expression on the left is greater than the expression on the right.

== Boolean test to see if the expression on the left is equal to the expression on the right.

<= Boolean test to see if the expression on the left is less than or equal to the expression on the right.

>= Boolean test to see if the expression on the left is greater than or equal to the expression on the right.

!= Boolean test to see if the expression on the left is not equal to the expression on the right.

AND, OR, NOT

Boolean operators used to join the result of multiple comparisons, or negate the given expression (what is *TRUE* becomes *FALSE*, and vice-versa)

SELECT ... END SELECT

Alternate form for a group of linear nested `IF` statements. Indent the sequence of statements surrounded by the SELECT and END SELECT.

```
SELECT single_variable
    CASE value1:
        ***block of statements***
    CASE value2:
        ***block of statements***
        .
        .
        .
    CASE valueN:
        ***block of statements***
    DEFAULT:
        ***block of statements***
END SELECT
```

Variable name construction guidelines: Names should be concise, but using complete words if possible. All letters should be lowercase; however, if a name is formed from multiple words, you may join the words together into one name by capitalizing the first letter of each word, except for the first word in the string (e.g. camelCasePhrase). Alternately, join words with underscores (_). Please do not use the dash (-), since it can be confused with the minus sign used in calculations.

CISP 300: Suggested Pseudocoding Keywords for Algorithms

Defining and using a record:

Record *RecordName*

Field declaration 1

Field declaration 2, etc

End Record

Declare *RecordName variableName*

Example:

```
Record Bowling
    String name
    Integer id
    Integer game1, game2, game3
    Integer totalScore
End Record

Declare Bowling player1

Display "Enter name: "
Input player1.name
Display "Enter id: "
Input player1.id
Display "Enter game 1 score: "
Input player1.game1
Display "Enter game 2 score: "
Input player1.game2
Display "Enter game 3 score: "
Input player1.game3
player3.totalScore = player1.game1 + player1.game2 + player1.game3
Display "Bowler, id, game1, game2, game3, total"
Display player1.name, player1.id, player1.game1, player1.game2, player1.game3,
player1.total
```