Pseudocode Definitions and Examples

| | |
|---|---|
| CONSTANT | Refers to creating a "special variable" that is a named constant with a set value that can not be changed during the program's execution. (Tip: Use this statement instead of DECLARE for variables whose value can not be changed)<br>Example: CONSTANT real interest_rate = 3.9 |
| DECLARE<br>Declare | Refers to creation of variable in memory. Variables must be created before they can be used. ( Tip: Data can not be stored in a memory location that hasn't been created, so therefore doesn't exist. ) |
| | Example : **DECLARE** ***datatype variable name***<br>DECLARE real weight<br>Example : DECLARE datatype variable name = value<br>DECLARE real weight = 0.0 |
| DISPLAY<br>Display | Refers to output that is DISPLAYED on the console or screen. Programming languages may use keyword such as WRITE, PRINT etc. to indicate output . |
| | Example : DISPLAY " string literal would be written here between the double quotes"<br>DISPLAY " What is your first name"<br>Example : DISPLAY variable<br>DISPLAY firstname |
| INPUT | Refers to the data that has been read when entered. |
| | Example: INPUT variable<br>INPUT firstname |
| SET<br>(optional) | Refers to an assignment statement; assignment of data to be stored in a variable. Can be the result of a calculation that is created with math operators. |
| | Example: SET grossPay = hoursWorked * payRate<br>Example: grossPay = hoursWorked * payRate |

**Rules and Important Terms**

| | |
|---|---|
| **Variable Names** | No spaces allowed<br>No punctuation allowed<br>Can not begin with a number<br>Must be descriptive of data<br>Can use underscore (Example: gross_pay)<br>Can use camel casing (Example: grossPay)<br>(Tip: All variable names are case sensitive (Example: Total is a different variable than total) |
| **Data Type** | **Type of data being stored**<br>Real (decimal data - Example: 3.26)<br>Integer (whole number – Example: 10)<br>String ( sequence of characters or a single character – Example: Yes or yes)<br>(Tip: Programming languages have many different datatypes. For example a s single character may have the datatype Char (or character) to store a single character. |
| **String Literal** | String that appears in the actual code of a program.<br>DISPLAY " This is a string literal" |
| **Math Operators** | **Addition +, Subtraction -, Multiplication *, Division /, Modulous MOD, Exponent ^** |
| **Numeric Literal** | Number that appears in the actual code of your program<br>Example : hourly_Rate = 10.00 |

## Problem :

Calculate a baseball players batting average. The formula is

Batting Average = Hits / Times at Bat

**TIP: Identify the I – P – O first by asking these questions:**

**What does the output need to be?**
The output needs to be the player's batting average

**What is the INPUT that we are given to work with?**
The number of hits
The number of times at bat

**What is the process ?**
The process to find batting average is to divide number of hits by number of times at-bat

**Algorithm: (informal instructions with a very particular and logical sequence of steps to solve the problem)**

1. **Get the data from the enduser by prompting for the number of hits and the number of times at-bat**
2. **Calculate the batting average by dividing the number of hits by the number of times at-bat**
3. **Display the output (value of batting average) on the screen.**

## Pseudocode:

1. **// This program will calculate batting average**
2. **// Create the variables**
3. DECLARE integer num_times_atbat
4. DECLARE integer hits
5. DECLARE real battingAverage
6. **//Prompt the user for input**
7. DISPLAY " Enter the number of times at bat"
8. INPUT num_times_atbat
9. DISPLAY " Enter the number of hits"
10. INPUT hits
11. **// Calculate the batting average**
12. SET battingAverage = hits / num_times_atbat
13. **//Display the, hits, times at bat and the batting average**
14. DISPLAY " The number of times at bat was" num_times_atbat
15. DISPLAY "The number of hits was" hits
16. DISPLAY " The batting average is" battingAverage