PRIMM Lesson Ideas
by Evans Chikasa
Written by teachers, for teachers

- Learning programming
- Counting items meeting a criteria
- Secret number guessing game
The algorithm below shows a program that performs an arithmetic operation based on some conditions.

 DECLARE  Number1 : INTEGER
 DECLARE  Number2 : INTEGER
 DECLARE  Number3 : INTEGER
 DECLARE  Number4 : REAL
 Number1 ← 0
 Number2 ← 0
 FOR  Count ← 1 TO 7
    INPUT  Number3
    IF  Number3 = -999 THEN
        IF  Number1 = 0 THEN
            OUTPUT "You cannot divide by 0"
        ELSE
            Number4 = Number2 / Number1
            OUTPUT Number4
        END IF
    ELSE
        Number1 ← Number1 + 1
        Number2 ← Number2 + Number3
        END IF
    END IF
 NEXT  Count
Predict:

Predict the outcome if the user inputs the following series of numbers.
1. 20
2. 25
3. 30
4. -15
5. -20
6. 20
7. -999

Run:

Work with a partner to use each number from the Predict stage and check if your predictions are correct. Use the table below to record all inputs and outputs.

<table>
<thead>
<tr>
<th>Number1</th>
<th>Number2</th>
<th>Number3</th>
<th>OUTPUT Number4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Investigate:

Discuss with your partner how the algorithm meets the requirements by identifying the following:

1. How the data is input?
2. How does the program decide which action to perform based on the comparison?
3. What criteria allow the program to terminate?
4. What criteria prevent division by zero?
The algorithm needs to be changed so that only numbers between 1 and 100 are input. Any number input outside this range should stop the algorithm and print the output. The algorithm should not allow division by zero.

Work with your partner to:
1. Identify which part of the algorithm needs to be changed.
2. Identify the change that needs to be made.
3. Re-write the algorithm with the change.
4. Select 6 sets of test data to check if your algorithm works. Create a table to record all your inputs, processes, and outputs.

Challenge

This algorithm can be represented as a flowchart. Discuss with your partner the different flowchart symbols you would need to translate the above pseudocode to flowchart. If you feel excited, challenge yourself to translate the new algorithm version into a working program using a programming language such as Python, VB.NET or JAVA. You can use the internet for any research on any aspect of the program you do not understand or need clarity.
Counting items meeting a criteria using PRIMM

The algorithm below shows a program that performs an arithmetic operation based on some conditions.

DECLARE Price : REAL
DECLARE TotalCost: INTEGER
DECLARE CountExpensive : INTEGER
DECLARE CountCheap: INTEGER
DECLARE itemCount: INTEGER

CountExpensive ← 0
CountCheap ← 0
TotalCost ← 0

FOR itemCount ← 1 TO 10
INPUT Price
    IF Price = -999 THEN
        OUTPUT "The total cost of all items is", TotalCost
        OUTPUT "The number of cheap items is", CountCheap
        OUTPUT "The number of expensive items is", CountExpensive
    ELSE
        IF Price > 150 THEN
            CountExpensive ← CountExpensive + 1
        ELSE
            CountCheap ← CountCheap + 1
        END IF
    END IF
    TotalCost ← TotalCost + Price
END IF
NEXT itemCount
### Predict:

Predict the output if the user inputs the following series of numbers.
- 200
- 250
- 30
- 150
- 2000
- 149.56
- 25.5
- -999

### Run:

Work with a partner to use each number from the Predict stage and check if your predictions were correct. Use the table below to record all inputs and outputs.

<table>
<thead>
<tr>
<th>itemCount</th>
<th>Price</th>
<th>CountExpensive</th>
<th>CountCheap</th>
<th>TotalCost</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Investigate:

Discuss with your partner how the algorithm meets the requirements by identifying the following:

1. How the data is input
2. How does the program decide which action to perform based on the comparison?
3. What criteria allow the program to terminate?
4. What criteria prevent division by zero?

Modify:

The algorithm needs to be changed so that only numbers between 1 and 100 are inputted. Any number input outside this range should stop the algorithm and print the output. The algorithm should not allow division by zero.

Work with your partner to:
1. Identify which part of the algorithm needs to be changed.
2. Identify the change that needs to be made.
3. Re-write the algorithm with the change.
4. Select 6 sets of test data to check if your algorithm works. Create a table to record all your inputs, processes, and outputs.

Make:

This algorithm can be represented as a flowchart.
1. Discuss with your partner the different flowchart symbols you would need to translate the above pseudocode to flowchart.

If you feel excited,
2. Challenge yourself to translate the new algorithm version into a working program using a programming language such as Python, VB.NET or JAVA. You can use the internet for any research on any aspect of the program you do not understand or need clarity.
Secret Number Guessing Game using PRIMM

The function \texttt{RAND(x)} generates a random decimal (real) number between 0 and x (not inclusive of x). For example, \texttt{RAND(5)} generates a random number between 0 and 5 (excluding 5), e.g., 4.3 or 2.1. On the other hand, the function \texttt{INT(y)} converts a decimal (real) number to an integer or whole number. For example, \texttt{INT(4.3)} and \texttt{INT(2.1)}

2

Arun is developing a computer game to play with his classmates. He first designed the game using an algorithm. The Pseudocode below shows how this game works.

\begin{verbatim}
DECLARE SecretNumber: INTEGER
DECLARE GuessedNumber: INTEGER
DECLARE NumberOfGuesses: INTEGER
DECLARE Word: STRING

SecretNumber ← INT(RAND(10))
OUTPUT "Please guess the secret number:"
INPUT GuessedNumber
NumberOfGuesses ← 1

WHILE GuessedNumber <> SecretNumber DO
    IF GuessedNumber < SecretNumber THEN
        OUTPUT "Your guessed number is smaller than the secret number"
    ELSE
        OUTPUT "Your guessed number is larger than the secret number"
    END IF

OUTPUT "Please try to guess the secret number again:"
\end{verbatim}
INPUT GuessedNumber
NumberOfGuesses ← NumberOfGuesses + 1

END WHILE

IF NumberOfGuesses = 1:
    Word ← "guess"
ELSE
    Word ← "guesses"
END IF

OUTPUT
    “Congratulations!!! You Win! It took you “ & NUM_TO_STRING(NumberOfGuesses) & ” “ & Word

Predict:
The program generates the SecretNumber as 6.
1 What should be output if the user enters the secret number 1?
2 What should be output if the user enters the secret number 5?
3 What should be output if the user enters the secret number 10?
4 What should be output if the user enters the secret number 7?
5 What should be output if the user enters the secret number 6?

Run:
Work with a partner to use each number from the Predict stage and check if your predictions were correct. Use the table below to record all inputs and outputs.

<table>
<thead>
<tr>
<th>SecretNumber</th>
<th>GuessedNumber</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Investigate:
Discuss with your partner how the algorithm meets the requirements by identifying the following:
1. How the data is input
2. How the input is compared to the secret number
3. What output or action is performed by the computer based on the results of the comparison?
4. How does the program decide whether to use the word “Guess” or “Guesses”? Why is this necessary?

Modify:
The algorithm needs to be changed to generate the secret number between 1 and 20 inclusive.

Work with your partner to:
1. Identify which part of the algorithm needs to be changed.
2. Identify the change that needs to be made.
3. Re-write the algorithm with the change.
4. Select and use a range of test data to check if your algorithm works.

If your algorithm does not work the first time, that is ok! Repeat the process until it does work.

Make:
Challenge yourself to translate the new algorithm version into:
1. A flowchart showing all the paths the algorithm will take before printing the final output of the program.
2. A working program using a programming language such as Python, VB.NET or JAVA. You can use the internet for any research on any aspect of the program you do not understand or need clarity.