Anslem Inomwan

Ivy Tech Community College

SDEV120- Computer Logic

Professor James R. Rowings, J.D.

November 11th, 2024

**DEBUG EXECRISE 1**

In reviewing and updating a program, I can follow a methodical approach to identify any issues in the code. The first step would be to carefully read through the pseudocode or the actual code to ensure that all syntax and logic are correct. I can check for common mistakes such as missing input statements, incorrect logic in conditional structures (like the while loop or if-else conditions), and any uninitialized variables.

For example, in the given pseudocode, there is a missing input statement for the firstTest variable before the while loop. This would need to be corrected to ensure the program behaves as expected.

When debugging, I can also test the program by providing different inputs to make sure that all potential scenarios are handled correctly. This includes testing boundary conditions, like entering 0 to exit the while loop.

To minimize errors, the developer could have taken several actions, such as thoroughly reviewing the pseudocode and ensuring all steps are logically connected. They could also have implemented clear input and output prompts to avoid confusion, and used parentheses around expressions to ensure proper order of operations. Additionally, testing the program with a variety of inputs beforehand could have helped catch errors before they became problematic.

I can improve readability and accuracy by reviewing the conditionals and calculations in the pseudocode. For example, ensuring the average calculation follows the correct formula, i.e., average = (firstTest + secondTest) / 2, with proper use of parentheses, is crucial for accurate results.

By following a structured debugging process, such as testing, verifying logic, and making necessary corrections, I can ensure that the code functions correctly and handles all edge cases appropriately.

**DEBUG EXECRISE 2 (Rewritten pseudocode)**

start

Declarations

string name

num hours

num rate

num DEDUCTION = 45

string EOFNAME = "ZZZ"

num gross

num net

output "Enter first name or ", EOFNAME, " to quit"

input name

while name not equal to EOFNAME

output "Enter hours worked for ", name

input hours

output "Enter hourly rate for ", name

input rate

gross = hours \* rate

net = gross - DEDUCTION

if net > 0 then

output "Net pay for ", name, " is ", net

else

output "Deductions not covered. Net is 0."

endif

output "Enter next name or ", EOFNAME, " to quit"

input name

endwhile

output "End of job"

stop

**DEBUG EXECRISE 3 (Rewritten pseudocode)**

start

Declarations

num firstTest

num secondTest

num average

num PASSING = 60

output "Enter first score or 0 to quit "

input firstTest

while firstTest not equal to 0

output "Enter second score "

input secondTest

average = (firstTest + secondTest) / 2

output "Average is ", average

if average >= PASSING then

output "Pass"

else

output "Fail"

endif

output "Enter first score or 0 to quit "

input firstTest

endwhile

Stop

**DEBUG EXECRISE 4 (Rewritten pseudocode)**

start

Declarations

num mortgagePayment

num utilities

num taxes

num upkeep

num total

startUp()

while mortgagePayment not equal to 0

mainLoop()

endwhile

finishUp()

stop

startUp()

output "Enter your mortgage payment or 0 to quit"

input mortgagePayment

return

mainLoop()

output "Enter utilities"

input utilities

output "Enter taxes"

input taxes

output "Enter amount for upkeep"

input upkeep

total = mortgagePayment + utilities + taxes + upkeep

output "Total cost of home ownership is ", total

output "Enter your mortgage payment or 0 to quit"

input mortgagePayment

return

finishUp()

output "End of program"

return

**Formulating and Implementing Your Testing Plan**:

* This involves designing a structured approach to test the pseudocode or program, ensuring all logic flows are covered. This includes considering edge cases, normal cases, and any potential errors that could occur during execution. Creating a detailed testing plan helps confirm that all variables, inputs, and outputs are handled correctly.

**Thoroughly Verifying the Accuracy of Calculations and Solutions**:

* I tested each calculation for correctness. For example, in the pseudocode, the net pay calculation and the average test score must be evaluated to ensure that they are correctly implemented. This includes checking for issues like incorrect order of operations, missing parentheses, or division by zero, as well as confirming that all logical conditions (e.g., for passing or failing) behave as expected.

**Methodically Executing the Test by Reviewing Each Line of Code**:

* After testing the program, carefully go through the code to identify any potential errors. This includes checking for spelling mistakes in variable names (like ouput instead of output), verifying logical conditions in loops and conditionals, and confirming that each part of the pseudocode is functioning as intended. It's also important to ensure the code follows good programming practices, such as proper indentation and clear documentation.