Programming Logic & Design

Chapter 7 – Input Validation

Queen's College

CSD 1133 – FSDM -2023S

Chapter 7 – Input Validation

One of the most famous sayings among computer programmers is "garbage in, garbage out." This saying, sometimes abbreviated as GIGO, refers to the fact that computers cannot tell the difference between good data and bad data. If a user provides bad data as input to a program, the program will process that bad data and, as a result, will produce bad data as output.

```
// Variables to hold the hours worked, the
// hourly pay rate, and the gross pay.
 Declare Real hours, payRate, grossPay
 // Get the number of hours worked.
 Display "Enter the number of hours worked."
 Input hours
 // Get the hourly pay rate.
 Display "Enter the hourly pay rate."
Input payRate
 // Calculate the gross pay.
 Set grossPay = hours * payRate
// Display the gross pay.
 Display "The gross pay is ", currencyFormat(grossPay)
```

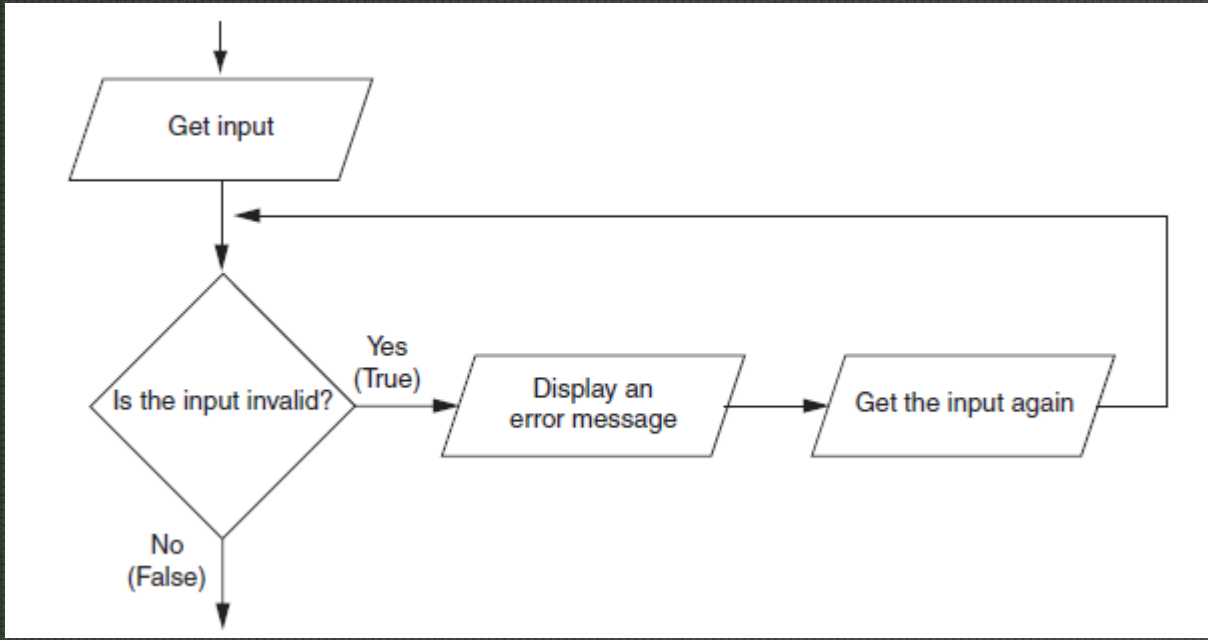
Program Output (with Input Shown in Bold)

```
Enter the number of hours worked.
400 [Enter]
Enter the hourly pay rate.
20 [Enter]
The gross pay is $8,000.00
```

Chapter 7 – Input Validation

Input Validation Loop

CONCEPT: Input validation is commonly done with a loop that iterates as long as an input variable contains bad data



The above program reads input in two places: first just before the loop and then inside the loop. The first input operation—just before the loop—is called a *priming read*, and its purpose is to get the first input value that will be tested by the validation loop. If that value is invalid, the loop will perform subsequent input operations.

Chapter 7 – Input Validation

```
// Get a test score.
Display "Enter a test score."
Input score

// Make sure it is not less than 0.
While score < 0
   Display "ERROR: The score cannot be less than 0."
   Display "Enter the correct score."
   Input score
End While
```

This pseudocode only rejects negative test scores. What if you also want to reject any test scores that are greater than 100? Modify the pseudocode to reject test scores greater than 100

In Chapter 5 we saw a program that Samantha can use to calculate the retail price of an item in her import business. Samantha has encountered a problem when using the program, however. Some of the items that she sells have a wholesale cost of 50 cents, which she enters into the program as 0.50.Because the 0 key is next to the key for the negative sign, she sometimes accidentally enters a negative number. She has asked you to revise the program so it will not allow a negative number to be entered for the wholesale cost. You decide to add an input validation loop to the showRetail module that rejects any negative numbers that are entered into the wholesale variable. Program in the next slide shows the new pseudocode, with the new input validation code shown in lines 28 through 33.

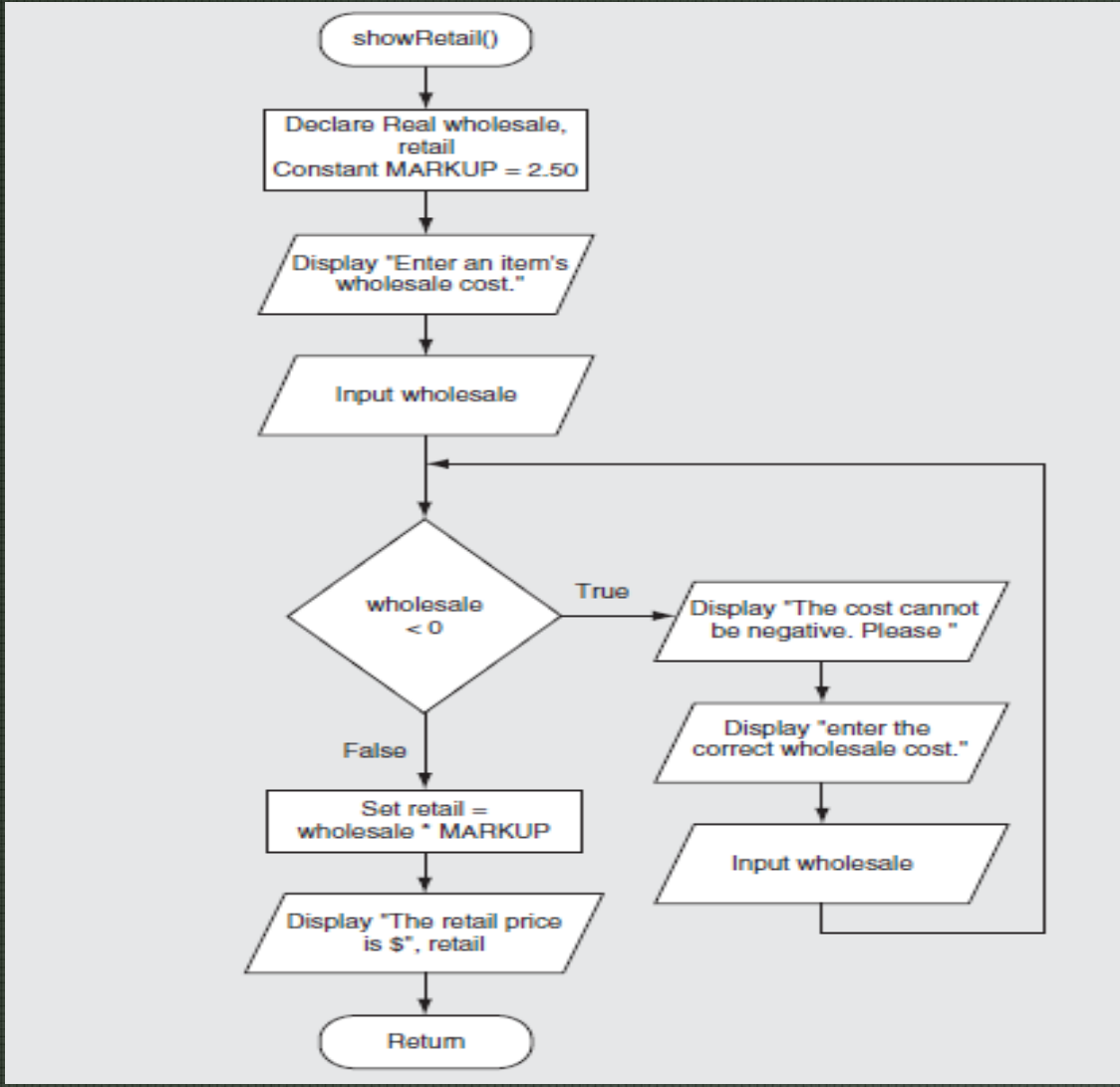
Chapter 7 – Input Validation

```
1 Module main()
2    // Local variable
3    Declare String doAnother
4
5    Do
6       // Calculate and display a retail price.
7       Call showRetail()
8
9       // Do this again?
10      Display "Do you have another item? (Enter y for yes)"
11      Input doAnother
12   While doAnother == "y" OR doAnother == "Y"
13 End Module
14
15 // The showRetail module gets an item's wholesale cost
16 // from the user and displays its retail price.
17 Module showRetail()
18    // Local variables
19    Declare Real wholesale, retail
20
21    // Constant for the markup percentage
22    Constant Real MARKUP = 2.50
23
24    // Get the wholesale cost.
25    Display "Enter an item's wholesale cost."
26    Input wholesale
27
28    // Validate the wholesale cost.
29    While wholesale < 0
30       Display "The cost cannot be negative. Please"
31       Display "enter the correct wholesale cost."
32       Input wholesale
33    End While
34
35    // Calculate the retail price.
36    Set retail = wholesale * MARKUP
37
38    // Display the retail price.
39    Display "The retail price is $", retail
40 End Module
```

Program Output (with Input Shown in Bold)

```
Enter an item's wholesale cost.
-0.50 [Enter]
The cost cannot be negative. Please
enter the correct wholesale cost.
0.50 [Enter]
The retail price is $1.25
Do you have another item? (Enter y for yes)
n [Enter]
```

Chapter 7 – Input Validation



Chapter 7 – Input Validation

Using a Posttest Loop to Validate Input

You might be wondering whether you could use a posttest loop to validate input instead of using the priming read. For example, the pseudocode to get a test score and validate it could be written as follows with a Do-While loop.

Do
Display "Enter a test score."
Input score
While score < 0 OR score > 100

Discuss the problem in the above code

It is usually better to have a priming read followed by a pretest validation loop.

```
// Get the model number.
Display "Enter the model number."
Input model

While model != 100 AND model != 200 AND model != 300
   Display "The valid model numbers are 100, 200, and 300."
   Display "Enter a valid model number."
   Input model
End While
```

you can simplify the validation loop by writing a Boolean function to test the model variable and then calling that function in the loop

Chapter 7 – Input Validation

```
// Get the model number.
Display "Enter the model number."
Input model

While isInvalid(model)
   Display "The valid model numbers are 100, 200, and 300."
   Display "Enter a valid model number."
   Input model
End While
```

function isInvalid() is given below

```
Function Boolean isInvalid(Integer model)
   // A local variable to hold True or False.
   Declare Boolean status

   // If the model number is invalid, set status to True.
   // Otherwise, set status to False.
   If model != 100 AND model != 200 AND model != 300 Then
      Set status = True
   Else
      Set status = False
   End If

   // Return the test status.
   Return status
End Function
```

Chapter 7 – Input Validation

Validating String Input

In some programs you must validate string input. For example, suppose you are designing a program that asks a yes/no question, and you want to make sure that only the strings "yes" or "no" are accepted as valid input. The following pseudocode shows how this might be done.

```
// Get the answer to the question.
Display "Is your supervisor an effective leader?"
Input answer
// Validate the input.
While answer != "yes" AND answer != "no"
   Display "Please answer yes or no. Is your supervisor an"
   Display "effective leader?"
   Input answer
End While
```

Why this design is too rigid? Can you use library function to improve the above code ?

Hint : use toLower() or toUpper()

Chapter 7 – Input Validation

Defensive Programming

- CONCEPT: Input validation is part of the practice of defensive programming. Thorough input validation anticipates both obvious and unobvious errors.

Defensive programming is the practice of anticipating errors that can happen while a program is running, and designing the program to avoid those errors. All of the input validation algorithms examined in this chapter are examples of defensive programming

Chapter 7 – Input Validation

Checkpoint

Defensive programming is the practice of anticipating errors that can happen while a program is running, and designing the program to avoid those errors. All of the input validation algorithms examined in this chapter are examples of defensive programming

- What does the phrase “garbage in, garbage out” mean?
- Give a general description of the input validation process.
- Describe the steps that are generally taken when an input validation loop is used to validate data.
- What is a priming read? What is its purpose?

Chapter 7 – Input Validation

Programming Questions

1. Payroll Program with Input Validation

Design a payroll program that prompts the user to enter an employee's hourly pay rate and the number of hours worked. Validate the user's input so that only pay rates in the range of $7.50 through $18.25 and hours in the range of 0 through 40 are accepted. The program should display the employee's gross pay.

2.Speeding Violation Calculator

Design a program that calculates and displays the number of miles per hour over the speed limit that a speeding driver was doing. The program should ask for the speed limit and the driver's speed. Validate the input as follows:

- The speed limit should be at least 20, but not greater than 70.
- The driver's speed should be at least the value entered for the speed limit (otherwise the driver was not speeding).Once correct data has been entered, the program should calculate and display the number of miles per hour over the speed limit that the driver was doing.