Input Validation

Learning Objectives

Garbage In, Garbage Out

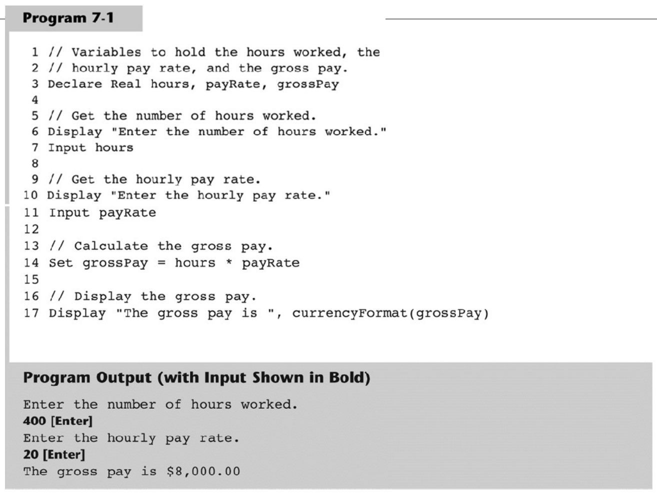
The Input Validation Loop

Defensive Programming

Garbage In, Garbage Out (1 of 2)

If a program reads bad data as input, it will produce bad data as outpu
- Programs should be designed to accept only good data
- Input Validation
  - All input should be inspected before processing
  - If it's invalid, it should be rejected and the user should be prompte to enter the correct data

Garbage In, Garbage Out (2 of 2)



Program 7-1

```
// 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)
```

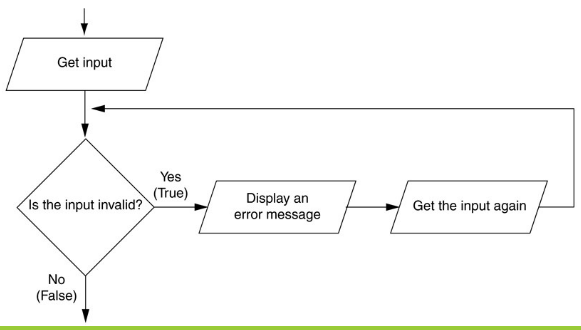
Program Output (with Input Shown in Bold)

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

The Input Validation Loop (1 of 3)

Input validation is commonly done with a loop that iterates as long as input is bad

Figure 7-1 Logic containing an input validation loop



The Input Validation Loop (2 of 3)

Priming read is the first input to be tested

```
// Get a test result
Display "Enter a test score."
Input score
//Make sure it is not lower than 0.
While score < 0 OR score > 100
    Display "ERROR:  The score cannot be less than 0 "
    Display "or greater than 100."
    Display "The the correct score."
    Input score
End While
```

The Input Validation Loop (3 of 3)

Writing Validation Functions
- For complex validation, it is recommended to write a function.
- This process can make the code look cleaner

Validating String Input
- Some strings must be validated such as those programs that ask for a specific string input like “yes”
- Or programs that specify a string to be a specific length like password validation

Defensive Programming (1 of 2)

Input validation is defensive programming
- The practice of anticipating both obvious and unobvious errors that can happen

Types of errors to consider
- Empty input, where a user accidentally hits enter before entering data
- The user enters the wrong type of data

Defensive Programming (2 of 2)

Common errors to be aware of

- State abbreviations should be 2-character strings
- Zip codes should be in the proper format of 5 or 9 digits
- Hourly wages and salary amounts should be numeric values and within ranges
- Dates should be checked
- Time measurements should be checked
- Check for reasonable numbers