B100/I101

Problem Solving Using Computers

Introduction to
Software Development and Problem Solving
Part 2

Adapted from Drs. Surma, Adaikkalavan, Hakimzadeh and Zhang
Three Problem Solving Steps

1) Understand the Question/Problem

2) Identify following:
   1) Inputs
   2) Outputs
   3) Equations

3) Any Limits/Constraints
Software Development Life Cycle, 6 steps

First two steps are part of the problem solving techniques

1) Requirement Specification
   ● Problem definition
     - Is the problem complete and clear?
     - What are the assumptions?

2) Analysis
   ● Input format
   ● Output format
   ● Requirements (formulas, facts, figures, etc)
   ● Constraints (limits, etc.)
   ● Identity possible solutions
Software Development Life Cycle, 6 steps

3) Design

- Develop a list of steps (an algorithm) to solve the problem
- An algorithm is often written in a generic language called pseudocode
- Desk check your algorithm

4) Implementation

- Convert the algorithm developed in the Design phase into a desired programming language (i.e. VB, C++)
- Writing comments and following accepted programming styles
Software Development Life Cycle, 6 steps

5) Testing
   ● Verify the correctness of your program (component testing, and overall testing)

6) Maintenance
   ● Add, modify, and maintain the system
Writing an Algorithm

- What is an Algorithm
  - A step-by-step solution to problem
  - Can be anything similar to:
    - 1-2-3 or A-B-C step approach
    - Outline
    - Do this then that then that

- Use as standard handout given out:
  - “An informal language for writing algorithms”
More Sample Problems

- Calculating student grades
- Find the smallest and largest Integer value entered by the user
Sample Problem: Calculating Student Grades

- Step 1: Problem definition
  - Write a program which accepts a student name and test score as input. The program then determines the letter grade for the student and produce an output similar to the following:

<table>
<thead>
<tr>
<th>Name</th>
<th>Score</th>
<th>Letter Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>95</td>
<td>A</td>
</tr>
</tbody>
</table>

- The letter grade are calculated as follows:
  - >=90  A
  - >=80  B
  - >=70  C
  - >=60  D
  - <60   F

- The test score is between 0 and 100
Sample Problem: Calculating Student Grades

- **Step 2: Analysis**
  - Determine the **input and output** of the program:
    - Input: student name and score
    - Output: student name, score and letter grade
  - Determine the **formulas, fact, etc. Needed**:
    - If score is higher than or equal to 90, A
    - If score is higher than or equal to 80 but lower than 90, B
    - If score is higher than or equal to 70 but lower than 80, C
    - …
  - Determine **constraints**:
    - Score must be between 0 and 100.
      - What if user inputs a value outside of this range?
  - Determine **possible solutions**:
    - Use (multiple) comparisons
Sample Problem: Calculating Student Grades

- Step 3: Design
  1) Read the student name and test score (testScore)
  2) Determine the letter grade (letterGrade)
     - if testScore >= 90
       letterGrade = “A”
     - if testScore >= 80 and testScore < 90
       letterGrade = “B”
     - if testScore >=70 and testScore < 80
       letterGrade = “C”
     - if testScore >=60 and testScore < 70
       letterGrade = “D”
     - if testScore < 60
       letterGrade = “F”
  3) Display the student name, testScore, and letterGrade
Module Module1
    Sub Main()
        Dim name As String
        Dim testScore As Double
        Dim letterGrade As String
        
        Console.Write("Please enter the student name? ")
        name = Console.ReadLine()
        
        Console.Write("Please enter the student test score? ")
        testScore = Console.ReadLine()
        
        If (testScore >= 90) Then
            letterGrade = "A"
        End If
        If (testScore >= 80) And (testScore < 90) Then
            letterGrade = "B"
        End If
        If (testScore >= 70) And (testScore < 80) Then
            letterGrade = "C"
        End If
        If (testScore >= 600) And (testScore < 70) Then
            letterGrade = "D"
        End If
        If (testScore < 60) Then
            letterGrade = "F"
        End If
        
        Console.WriteLine("Name     Score       Letter Grade")
        Console.WriteLine("{0}     {1}         {2}", name, testScore, letterGrade)
        
        Console.ReadLine() 'just to pause the program
    End Sub
End Module
Sample Problem: Find Largest

- **Problem Definition**
  - The user is asked to enter a few positive numbers
  - If the user enters a zero, the program should stop
  - After the program stops, it should display the largest numbers entered by the user

- **Analysis**
  - Determine the **input and output** of the program:
    - Input: a set of non-negative integer numbers
    - Output: the largest positive number
  - Determine **constraints**:
    - None?
  - Determine **possible solutions**???
Sample Problem: Find Largest

Analysis

- Determine the **input and output** of the program:
  - Input: a set of integer numbers
  - Output: the largest number

- Determine **constraints**:
  - no

- Determine **possible solutions**
  - “remember” the largest number so far, and compare it with each newly input number…
Sample Problem: Find Largest

- Design
  - Get a number from the user
  - Set the Largest to the above number
  - If the number is zero, stop and output the Largest
  - Else get a new number from the user
  - If the number is larger than Largest, replace the largest with the new number
  - If the number is zero, stop and output the Largest
  - Else get a new number from the user
  - If the number is larger than Largest, replace the largest with the new number
  - If the number is zero, stop and output the Largest
  - …
Sample Problem: Find Largest

- **Design** *(a better way, using loop!)*
  - Get a number from the user
  - Set the Largest to the above number
  - Start a loop which only stops when the user enters a zero
  - Inside the loop, check to see if the number is larger than the largest. If so, replace the largest with the new number
  - Inside the loop, get a new number from the user
  - After the loop, output the Largest
Alternatively, Present your Design using Pseudocode

```plaintext
read number
largest = number

while (number <> 0) do
    if (number > largest) then
        largest = number
    end if
    read number
end while

write "The largest was", largest
```
Pseudocode

- An informal language that helps you develop algorithms
- Similar to everyday English
- Convenient and friendly
- But not an actual computer programming language
Or, you can “draw” your design!

```
Start

input number

largest = number

while number <> 0

if number > largest

largest = number

input number

else

output largest

End

Flow Chart
```
Convert your design into code

**Module** Module1

**Sub** Main()

```vbnet
Dim number, largest As Integer

Console.WriteLine("Enter a series of integer numbers (0 to stop)"
number = Console.ReadLine()
largest = number

While (number <> 0)
    If (number > largest) Then
        largest = number
    End If
    number = Console.ReadLine()
End While

Console.WriteLine("The largest was " & largest)
Console.ReadLine() 'pause the program
```

**how about find the smallest?**

```vbnet
read number
largest = number

while (number <> 0) do
    if (number > largest) then
        largest = number
    end if
    read number
end while

write "The largest was", largest
```
Find the smallest and largest

Module Module1

Sub Main()
    Dim number, smallest, largest As Integer

    Console.WriteLine("Enter a series of integer numbers (0 to stop)")
    number = Console.ReadLine()
    smallest = number
    largest = number

    While (number <> 0)
        If (number < smallest) Then
            smallest = number
        End If
        If (number > largest) Then
            largest = number
        End If
        number = Console.ReadLine()
    End While

    Console.WriteLine("The smallest was " & smallest)
    Console.WriteLine("The largest was " & largest)
    Console.ReadLine() 'just to pause the program

End Sub

End Module