I101/B100
Problem Solving with Computers

By:

Dr. Hossein Hakimzadeh
Computer Science and Informatics
IU South Bend
What skill are needed for software development?

- Problem Solving skills.
- People skills.
- Communication Skills.
- Programming Skills.
What is Problem Solving?

☐ A mental process.

☐ A creative process.

☐ Problem solving occurs when one needs to move from a given state to a desired goal state.

☐ Ability to think critically.
What is Programming:

- The act of writing a program.
- To write a program, we need a **programming language**.
What is a Program?

- A plan to achieve a solution to a problem.
- A set of sequential instructions, which cause a computer to perform a particular operation or task.
Programming Languages:

- Machine Language (1st. Generation Languages or 1GL’s)

- Low level languages (2nd GL’s or Assembly Language)

- High Level Languages (3rd GL’s or languages such as Basic, Pascal, C, C++, COBOL)

- 4GL's (Languages used for Rapid Application Development, or Languages build for a specialized application)
Machine Language

- Based on 0's and 1's
- Instructions are written in 0's and 1's

Example:

```
   0110 10001 10100 10110
| \    \  \  \  \
ADD  X  Y  Z
```
Low level (Assembly) Languages

- Low level or Assembly Languages provides a one-to-one mapping between symbols (names) and computer instructions and memory locations.

- Assembly languages are tightly coupled to the machine (processor).

- Example:
  - ADD X, Y, Z
High Level Languages

- Closer to English
- Basic, Fortran, C, C++, Java, Cobol, etc.
- Easier to write and debug programs.

Example:

- \[ Z = X + Y \]
- If \( Z \geq 100 \) then
  - console.writeline("Z is big")

- Any high level language must eventually be translated to machine language before the computer can understand it.
What is a Compiler?

- A program that translates a high level language into machine language.

Source Code
Or
Source Program
C++, VB, etc.

Compiler

Object Code
Or
Object Program
Steps in Writing, Compiling and Running a program:

- **Editor**
  - Write Program
  - `*.vb`

- **Compiler**
  - Compile the Program
  - `*.EXE`

- **Run the .EXE file**
Software Development Life Cycle

1) Requirement specification
   - Problem Definition

2) Analysis
   - Input and Output format
   - Requirements (formulas, facts, figures, etc..)
   - Constraints (limits, etc..)
   - Identify possible solutions

3) Design
   - Develop a list of steps (an Algorithm) to solve the problem.
   - An algorithm is often written in a generic language called pseudo-code.
   - Desk check your algorithm.

4) Implementation
   - Convert the algorithm developed in the Design phase into a desired programming language (i.e. VB).

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

6) Maintenance
   - Add, modify, and maintain the system.
Problem Solving Techniques

1) Ask questions to fully understand the problem
   ■ What is my data? (what does it look like?, How much data is there?)
   ■ How will I know when I have processed all the data?
   ■ What should the output look like?
   ■ How many times is the process going to be repeated?
   ■ What special error conditions might arise?

2) Identify patterns
   ■ Reuse previous solutions, do not reinvent the wheel

3) Solve by analogy
   ■ Think of similar problems and how you solved them.

4) Means end analysis
   ■ Given a set of input data, how do we reach the desired output results, provided a set of tools at our disposal.

5) Divide and conquer
   ■ Divide the problem into a series of smaller, more manageable problems.

6) Merging Solution
   ■ Some problems can be viewed as a combination of 2 or more existing problems. (merge the existing solutions to get the new solution!)

7) Start by starting
   ■ Start by rewriting the problem in your own words. Try explaining the problem to your friend.

8) Algorithmic Problem Solving
   ■ Develop a step by step solution to a problem and then refine it. (step wise refinement)
Sample Problems?

1. Compute and display the Area and the Circumference of the circle.

2. Calculating Student Grades

3. Find the Smallest and Largest Integer Value entered by the User
Problem 1:

Given the radius of a circle, compute and display the Area and the Circumference of the circle.
Problem 1: Given the radius of a circle, compute and display the Area and the Circumference.

Analysis:

☐ Determine the Input and Output of the program

☐ Determine the formulas, fact, etc. needed

Design:

☐ Develop a list of steps to solve the problem (An algorithm)
Problem 1: Given the radius of a circle, compute and display the Area and the Circumference.

Analysis:
- Determine the Input and Output of the program:
  - Radius of a circle

- Determine the formulas, fact, etc. needed:
  - Area = PI * (Radius)$^2$
  - Circumference = 2 * PI * Radius
  - PI = 3.14159

Design:
- Develop a list of steps to solve the problem (An algorithm)

  1) read the radius
  2) compute the Area
     2.1) Area = PI * (Radius)$^2$
  3) computer the circumference
     3.1) Circumference = 2 * PI * Radius
  4) print the area and circumference
Problem 1: Given the radius of a circle, compute and display the Area and the Circumference.

Implementation:

```vbnet
Module Module1
    Sub Main()
        Dim radius, area, circumference As Double
        Const PI = 3.14159

        Console.Write("Please enter the radius of the circle? ")
        radius = Console.ReadLine()
        area = PI * radius * radius
        circumference = 2 * PI * radius
        Console.WriteLine("Area = " & area)
        Console.WriteLine("Circumference = " & circumference)

        Console.ReadLine() 'just to pause the program.
    End Sub
End Module
```
Problem 2:
Calculating Student Grades
Problem 2: Calculating Student Grades

- Write a program which accepts a student name and test score as input. The program then determines the letter grade for the student and produces 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>93</td>
<td>A</td>
</tr>
</tbody>
</table>

- The Letter Grades are calculated as follows:
  - >=90  A
  - >=80  B
  - >=70  C
  - >=60  D
  - < 59  F

- The test score is between 1 and 100.
Problem 2: Calculating Student Grades

Problem Definition:
- Do you understand the problem? If so write it in your own words

Analysis:
- Determine the Input and Output of the program
- Determine the formulas, fact, etc.

Design:
- Develop a list of steps to solve the problem (An Algorithm)

Implementation:
- Convert your design into VB code
**Problem 2: Calculating Student Grades**

**Design:**

1) read the student name and test_score
2) Determine the letter grade
   - if testscore >=90 and testscore <= 100
     letter grade = “A”
   - if testscore >=80 and testscore < 90
     letter grade = “B”
   - if testscore >=70 and testscore < 80
     letter grade = “C”
   - if testscore >=60 and testscore < 70
     letter grade = “D”
   - if testscore < 60
     letter grade = “F”
3) Display the name, test score and letter grade
Problem 2: Calculating Student Grades
Convert your design into code

```vbnet
Module Module1
    Sub Main()
        Dim name As String
        Dim test_score As Double
        Dim letter_grade As String

        Console.Write("Please enter the student name? ")
        name = Console.ReadLine()

        Console.Write("Please enter the student test score? ")
        test_score = Console.ReadLine()

        If (test_score >= 90) And (test_score <= 100) Then
            letter_grade = "A"
        End If
        If (test_score >= 80) And (test_score < 90) Then
            letter_grade = "B"
        End If
        If (test_score >= 70) And (test_score < 80) Then
            letter_grade = "C"
        End If
        If (test_score >= 60) And (test_score < 70) Then
            letter_grade = "D"
        End If
        If (test_score < 60) Then
            letter_grade = "F"
        End If

        Console.WriteLine("Name           Score       Letter Grade")
        Console.WriteLine("{0}            {1}         {2}", name, test_score, letter_grade)
        Console.ReadLine() 'just to pause the program.
    End Sub
End Module
```
Problem 3:

Finding the Smallest Integer Value entered by the User
**Problem 3:** Find the Smallest Integer value entered by the user

**Problem Definition:**
- Do you understand the problem? if so write it in your own words

**Analysis:**
- Determine the Input and Output of the program
- Determine the formulas, fact, etc.

**Design:**
- Develop a list of steps to solve the problem (An Algorithm)
Problem 3: Find the Smallest Integer value entered by the user

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 Smallest number entered by the user.

Analysis:
- Input should be integer numbers $\geq 0$.
- Each new number should be compared with the old Smallest number, to see if a new “Smallest” has been found.

Design:
- Get a number from the user.
- Set the Smallest to the above number
- Start a loop which only stops when the user enters a zero.
- Inside the loop get a new number from the user.
- Check to see if the number is smaller than the smallest. If so, replace the smallest with the new number.
Convert your Design into Pseudocode

- Number = Read from keyboard
- Smallest = Number
- loop until Number = 0

  - if Number < Smallest then
    - Smallest = Number.
    - Number = Read from keyboard

- End Loop
- Display the Smallest
Option Strict On
Option Explicit On

Module Module1

Sub Main()
    Dim Smallest, Number As Integer
    Console.WriteLine("Enter a series of positive integers. (0=stop)")
    Number = CInt(Console.ReadLine())
    Smallest = Number
    Do Until Number = 0
        If Number < Smallest Then
            Smallest = Number
        End If
        Number = CInt(Console.ReadLine())
    Loop
    Console.Write("The smallest was : ")
    Console.WriteLine(Smallest)
End Sub
End Module