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 clearly and critically.
Polya’s Problem Solving Techniques

- In 1945 George Polya published the book “How To Solve It” which quickly became his most prized publication.
- Came up with 4 principles:
  1. Understand the problem
  2. Devise a plan
  3. Carry out the plan
  4. Look back (reflect which helps you with future problems)
Computational Thinking

- Computational thinking is used to take a complex problem, understand the problem, and develop possible solutions.

**Basic Characteristics of computational thinking**

- **decomposition**
  - breaking down a complex problem into smaller, more manageable parts

- **Pattern recognition**
  - looking for similarities among and within a problem

- **Abstraction**
  - focusing on the important information only, ignoring irrelevant detail

- **Algorithm**
  - developing a step-by-step solution to the problem
Engineering Design Process

To solve engineering problems, engineers follow a series of steps called the "Engineering Design Process."

Five-Step Process:

- **ASK**: What is the problem? How have others approached it? What are your constraints?

- **IMAGINE**: What are some solutions? Brainstorm ideas. Choose the best one.

- **PLAN**: Draw a diagram. Make lists of materials you will need.

- **CREATE**: Follow your plan and create something. Test it out!

- **IMPROVE**: What works? What doesn't? What could work better? Modify your design to make it better. Test it out!

https://www.eie.org/overview/engineering-design-process
Pseudocode

- Before writing a program we often start by writing a rough draft using pseudocode.

- Pseudocode resembling a simplified programming language.

- The computer can’t run pseudocode instructions, but writing pseudocode allows a high-level look at the major steps in a program.

- Watch this videos:
  [https://www.youtube.com/watch?time_continue=54&v=HhBrkpTqzqg&feature=emb_logo](https://www.youtube.com/watch?time_continue=54&v=HhBrkpTqzqg&feature=emb_logo)
PSEUDOCODE:
Pseudocode is step-by-step written outline of your code that you can gradually transcribe into the programming language.

https://www.youtube.com/watch?v=N6uHKCbJEcE
Flow Chart

- A diagram that represents a workflow or process.

- A diagrammatic representation of an algorithm


https://www.edrawmax.com/flowchart/

https://www.visual-paradigm.com/tutorials/flowchart-tutorial/
Flowchart symbols

- Indicates **starting** point or **ending** point of the flowchart or algorithm.
- Indicates **input** to or **output** from the process within the algorithm.
- Indicates the **process** or operation that is performed within the algorithm.
- Indicates the **selection** or **decision** within the algorithm.

Flowchart for an algorithm to calculate difference in 2 numbers

**Calculate difference in 2 numbers**

1. Input number 1
2. Input number 2
3. If number 1 is equal to or bigger than number 2, then subtract number 2 from number 1, otherwise subtract number 1 from number 2
4. Store the answer as number 3
5. Output number 3

https://www.youtube.com/watch?v=Yq1OPs5hCt0
Some problems?

- Finding the smallest number in a pile of numbers.
- Computing the Area and Circumference of a circle
- Calculating Student Grades
- Finding the shortest path between two locations.
- Determining someone’s FICO score.
- Figuring how much paint you need to paint a room. What if the room is the shape of a cylinder or a ball?
- Figuring out how much grain fits in a silo.
- Figuring out how much money you will have for college if you save $10 per month for the next 10 years.
Problem 1:

Finding the Smallest Integer Value entered by the User
HOW DO WE FIND THE SMALLEST NUMBER?

Given five numbers (one at a time) would you be able to find or identify the smallest number?

• How does a Human approach this problem?

• How does a Human using a computer approaches this problem?
HOW DO WE FIND THE SMALLEST NUMBER?

Given five numbers (one at a time) would you be able to find or identify the smallest number?

88
HOW DO WE FIND THE SMALLEST NUMBER?

Given five numbers (one at a time) would you be able to find or identify the smallest number?

23
HOW DO WE FIND THE SMALLEST NUMBER?

Given five numbers (one at a time) would you be able to find or identify the smallest number?

88, 23, 712, 49, 14
HOW DO WE FIND THE SMALLEST NUMBER?

Given five numbers (one at a time) would you be able to find or identify the smallest number?

88, 23, 712, 49, 14
HOW DO WE FIND THE SMALLEST NUMBER?

Given five numbers (one at a time) would you be able to find or identify the smallest number?

14
1) WHAT WAS THE SMALLEST NUMBER?

?

2) WHAT STRATEGY DID YOU USE TO SOLVE THE PROBLEM?
   STRATEGY 1:
   STRATEGY 2:
   STRATEGY 3:

3) WAS YOUR STRATEGY “GOOD”? (SIMPLE, FAST, SCALABLE, ETC?)
   WILL YOUR STRATEGY WORK IF WE HAD 50 NUMBERS?
Problem 1: 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 1:** 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.
Problem Solving Using

Pseudocode & Flow chat
FLOWCHART AND PSEUDOCODE

Finding the Smallest Number entered by the user.
FIND THE SMALLEST NUMBER ENTERED BY THE USER

BASIC DESIGN

• Get a Number from the user.

• Set the Smallest to the above Number

• Repeat until the user tells us to stop!

  • get a new Number from the user.
  • check to see if the new Number is smaller than the smallest. If so, replace the smallest with the new number.
  • ask the user if they want to stop
  • repeat
Convert your Design into **Pseudocode**

```plaintext
Print “Please enter a number: ”
Number = Read from keyboard
Smallest = Number
StopLoop = NO

Repeat until StopLoop = YES
  Print “Do you want to stop? (YES or NO)”
  StopLoop = read from keyboard
  If StopLoop = NO
    Print “Enter another number: “
    Number = Read from keyboard
    if Number < Smallest then
      Smallest = Number

End of Loop

Print Smallest
```
Convert your Design into Flow Chart

Flow Chart created using: https://app.creately.com
From Problem Solving to Programming

Going from Problem Solving to Programming

- Converting your Pseudocode or Flowchart to a Program
Programming

- What is a program?
  - Set of instructions to do something.
three fundamental parts of all programming languages are: sequence, selection, and iteration.

https://www.youtube.com/watch?v=eSYeHlwDCNA&t=1s
Basic Programming Constructs:

- Variables
- Operators (math operators, assignment operator)
- Input, Output
- Control Structures
  - Sequencing
  - Selection (checking a condition)
  - Iteration (Looping)
- Modules
Module Module1

Sub Main()
    Dim Number, Smallest As Integer
    Dim StopLoop As String

    Console.Write("Please enter a number: ")
    Number = Console.ReadLine()
    Smallest = Number
    StopLoop = "NO"

    Do Until (StopLoop = "YES")
        Console.Write("Do you want to stop? (YES or NO): ")
        StopLoop = Console.ReadLine()
        If (StopLoop = "NO") Then
            If Number < Smallest Then
                Smallest = Number
            End If
        End If
        Console.Write("Please enter another number: ")
        Number = Console.ReadLine()
    End If
End Sub

End Module
VARIABLES:

- **Variables** are used by programmers to access and manipulate **memory**.
A variable is the **name given to a memory location**. (the memory location may hold data items such as numbers, characters, etc.)

- \( X = 5 \)
- Name = “Jack”
- Price = 55.87
BASIC ARITHMETIC OPERATORS

• Probably you have seen:
  • +, -, *, /

• Probably you have not seen:
  • MOD (%)  
  • TO the Power of (^)
The equal sign “=“

X = 5

This means:
• Copy 5 into variable X
• or
• Copy the right-hand side to the left-hand side
**ASSIGNMENT STATEMENT**

- **Syntax:**
  - `variable = expression`

- "=" is the assignment operator.

- An expression is a combination of variables, constants, numbers and operators.

- **Examples:**
  - `N1 = N2` 'variable = variable'
  - `N1 = 5` 'variable = constant number'
  - `N1 = N2 * 5 + 1` 'variable = expression'
  - `N1 = N1 + 5`
INPUT AND OUTPUT

• Input:
  • Name = read()

• Output:
  • Print “Hello”
  
  Or

  • Print Name
STRUCTURED PROGRAM THEOREM

- It states that any computer program can be written using only three control structures:
  - Sequence
  - Selection / Conditional
  - Iteration / Loop

Created by Deborah J. Hwang
SEQUENCE

• The default flow of program control is sequence.

• Basic computational statements of assignment, input, and output are executed sequentially, in the order encountered in a program.
• Selection control structures allow one path to be followed among several possible coding paths.
• Loop control structures allow a block of code to be repeated a number of times.
• The main method by which we solve large problems!
EVENTS & EVENT HANDLERS

- What is an event?
  - In computer programming, an **event** is a programmed action that occurs as a result of the user or another source, such as a mouse click.

- What is an event handler?
  - An **event handler** is the code that deals with the event, allowing a programmer to write code that will be executed when the event occurs.
Problem 2:

Given the radius of a circle, compute and display the Area and the Circumference of the circle.
Problem 2: 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 2: 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 2: Given the radius of a circle, compute and display the Area and the Circumference.

Implementation:

- 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 3:

Calculating Student Grades
Problem 3: 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 3: 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 3: Calculating Student Grades

Design:

- Read the student name
- Read the student testscore
- Determine the letter grade

\[
\begin{align*}
\text{if } \text{testscore } \geq 90 \text{ and } \text{testscore } \leq 100 \\
&\quad \text{letter grade } = \text{“A”} \\
\text{if } \text{testscore } \geq 80 \text{ and } \text{testscore } < 90 \\
&\quad \text{letter grade } = \text{“B”} \\
\text{if } \text{testscore } \geq 70 \text{ and } \text{testscore } < 80 \\
&\quad \text{letter grade } = \text{“C”} \\
\text{if } \text{testscore } \geq 60 \text{ and } \text{testscore } < 70 \\
&\quad \text{letter grade } = \text{“D”} \\
\text{if } \text{testscore } < 60 \\
&\quad \text{letter grade } = \text{“F”}
\end{align*}
\]

- Display the name, test score and letter grade
Problem 3: 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.WriteLine("Please enter the student name? ")
        name = Console.ReadLine()

        Console.WriteLine("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
```
How do we find the smallest number?

Given the following numbers: 7, 88, 23, 12, 49, 14

- How does a Human approach this problem?
- How does a Human using a computer approaches this problem?
Figuring out how much grain fits in a silo.

Given a silo with the with 10 feet diameter, 20 feet high, and a dome, how much grain can you fit in it?

☐ How does a Human approach this problem?

☐ How does a Human using a computer approaches this problem?
Some information you might need!

Volume of a Cylinder
\[ \pi r^2 h \]

Volume of Sphere
\[ \frac{4}{3} \pi r^3 \]

Volume of Cone
\[ \frac{1}{3} \pi r^2 h \]
Silo Volume Calculation (GUI)

![Silo Volume Calculator GUI](image)

- **Silo Diameter (feet)**
- **Silo Cylinder Height (feet)**
- **Dome/Silo Radius (feet)**

Buttons:
- Fill the Cylinder
- Fill the Dome

**Silo Volume (Cubic feet)**
Solution to the problem

- Once we develop a solution to the problem, we need to tell the computer about it!

- That requires that we speak in a language that a computer understands.
Public Class Form1

Const PI = 3.14159

Private Sub CalculateCylinderVolume_Click(sender As Object, e As EventArgs) Handles CalculateCylinderVolume.Click
If (SiloDiameter.Text = "") Or (SiloCylinderHeight.Text = "") Then
    If (SiloDiameter.Text = "") Then
        SiloDiameter.Text = ""
        SiloDiameter.BackColor = Color.Yellow
    End If
    If (SiloCylinderHeight.Text = "") Then
        SiloCylinderHeight.Text = ""
        SiloCylinderHeight.BackColor = Color.Yellow
    End If
Else
    SiloVolume.Text = CStr(PI * CDbl(SiloRadius.Text) ^ 2 * CDbl(SiloCylinderHeight.Text))
End If
End Sub

Private Sub CalculateCylinderPlusDome_Click(sender As Object, e As EventArgs) Handles CalculateCylinderPlusDome.Click
If (SiloDiameter.Text = "") Or (SiloCylinderHeight.Text = "") Then
    If (SiloDiameter.Text = "") Then
        SiloDiameter.Text = ""
        SiloDiameter.BackColor = Color.Yellow
    End If
    If (SiloCylinderHeight.Text = "") Then
        SiloCylinderHeight.Text = ""
        SiloCylinderHeight.BackColor = Color.Yellow
    End If
Else
    SiloVolume.Text = CStr(PI * CDbl(SiloRadius.Text) ^ 2 * CDbl(SiloCylinderHeight.Text) +
                          (4 / 3 * PI * CDbl(SiloRadius.Text) ^ 3) * 1 / 2)
End If
End Sub

Private Sub ButtonQuit_Click(sender As Object, e As EventArgs) Handles ButtonQuit.Click
    Close()
End Sub

Private Sub SiloDiameter_TextChanged(sender As Object, e As EventArgs) Handles SiloDiameter.TextChanged
If (SiloDiameter.Text <> "") Then
    SiloDiameter.BackColor = Color.White
    SiloRadius.Text = CStr(CDbl(SiloDiameter.Text) / 2)
End If
End Sub

Private Sub SiloCylinderHeight_TextChanged(sender As Object, e As EventArgs) Handles SiloCylinderHeight.TextChanged
If (SiloCylinderHeight.Text <> "") Then
    SiloCylinderHeight.BackColor = Color.White
End If
End Sub

Private Sub ButtonClear_Click(sender As Object, e As EventArgs) Handles ButtonClear.Click
    SiloCylinderHeight.Text = ""
    SiloDiameter.Text = ""
    SiloRadius.Text = ""
    SiloVolume.Text = ""
    SiloCylinderHeight.BackColor = Color.White
    SiloDiameter.BackColor = Color.White
End Sub

(c) Copyright 2022, H. Hakimzadeh
What is Programming:

☐ The act of writing a program.

☐ To write a program, we need a **programming language**.

☐ We also need an **algorithm**. (step by step instructions...)


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 (1\textsuperscript{st} Generation Languages or 1GL’s)

- Low level languages (2\textsuperscript{nd} GL’s or Assembly Language)

- High Level Languages (3\textsuperscript{rd} GL’s or languages such as Basic, Java, Pascal, C, C++, C#, PHP, COBOL, Fortran)
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             |

  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
  - \texttt{console.writeln("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)