Problem Solving Using Computers

Introduction to Software Development and Problem Solving

Part 1

Adapted from Drs. Adaikkalavan, Hakimzadeh, Surma and Zhang
What Skills 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 state.

- Needs ability to think critically.
  - Critical thinking clarifies goals, examines assumptions, discerns hidden values, evaluates evidence, accomplishes actions, and assesses conclusions.
Three Things to do first to solve problem.

1) Do you understand what the question ask?

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

3) Are there any constraints or limits
What is Programming?

- What is Programming?
  - Programming is problem solving.

- What is a program?
  - A set of sequential instructions, which cause a computer to perform a particular operation or task.
  - A program is used to solve a problem.

- To write a program, we need programming language
Three types of programming languages

1. Machine languages (Binary Language)
   - Strings of 0’s and 1’s
   - Understandable for machines (only)
   - Example:
     
     00110 10001 10100 10110
     
     \[
     \text{ADD} \quad \text{R1} \quad \text{R2} \quad \text{R3}
     \]
2. **Assembly languages** (or low level languages)

- Low level or Assembly Language provides a one to one mapping between a symbol (name) and computer instructions and memory locations.
- Tightly coupled to the machine language
- Example:

```
ADD R1, R2, R3
```
3. High-level languages

- Closer to English, Easier to write and debug programs.
- Use mathematical notations (translated via compilers)
- Basic, Fortran, C, C++, Pascal, Cobol, Java, etc..
- Example:

  ```java
  sum = partOne + partTwo
  If (sum >= 100) then
      console.writeline ("sum is big")
  ```

- Any high level language must eventually be translated to machine language before the computer can understand it.
Programming Languages

High-Level Languages

Assembly Languages

Machine Languages

Sum = partOne + partTwo

ADD R1, R2, R3

00110 10001
10100 10110
Writing, Compiling, and Running a Program

**Editor**
- **Write** Program – generate source file

**Compiler**
- **Compile** (and link) – generate executable file
- **.vb**
- **.EXE**

(Load and) **Run** the .EXE file
Compiler

- a computer program that translates a computer program written in one computer language (called the source language or source code) into an equivalent program written in another computer language (called the output or the target language, usually machine language or called the executable).
Compilers are computer programs written by someone else

- You can also write your own

**Translates** code from high-level to machine language

- **Source code**
  - The original program in a high level language
  - *Typically what you write*

- **Object code**
  - The translated version in machine language

There are many compilers for a single language

- ANSI standard for programming languages
Compiler

- **Capabilities**
  - Finds and reports *errors*. These errors can include incorrect syntax, misspelled keywords, and type mismatches.
  - e.g., *integre* instead of *integer*

- **Limitations**
  - When compiler find errors, the *user* must correct them and *recompile*
  - Compiler *will not* find *logic or semantic errors*
    - $Z = X + Y$ instead of $Z = X - Y$
We often use programs/routines written by other programmers
  - Those programs we use are already compiled
  - Their object code is available for us to use
  - Example: Input from keyboard, calculating square root

A **Linker** is also a program that combines
  - The object code for the programs we write and
  - The object code for the pre-compiled routines into
  - The machine language program (aka executable) the CPU can run
Load and Run

- Loader
  - is also a computer program
  - Copies the executable file into memory (RAM)
  - Initiates the execution of instructions

- Execution consists of
  - A program
  - Some input data
The Code Translation Hierarchy

Source Code (C++, VB, etc.)

Compiler

Assembler

Assembly Code

Object Code

Library Routines

Linker

Executable

Loader

Machine Code

Memory
About Visual Basic

- **BASIC** (Beginner’s All-purpose Symbolic Instruction Code)
  - Mid-1960’s at Dartmouth College for introducing novices and fundamental programming techniques
  - Microsoft’s first product: an *interpreter* for BASIC

- **Visual Basic**
  - Introduced by Microsoft in 1991 to make GUI-based Windows application development easier
  - VB programs are created using Visual Studio – IDE (Integrated Development Environment)
  - Later versions are fully object-oriented
  - Provides: Event driven programming
What is an algorithm?

- **Definition:** A set of clear ordered steps to solve a problem.
  - the **actions** to be executed and
  - the **order** in which these actions are to be executed

- **Examples:**
  - cooking recipes
  - learning to drive
  - setting a timer recording on the VCR
  - military procedures
  - computer programs
“Rise-and-shine algorithm” for a junior executive

1) Get out of bed
2) Take off pajamas
3) Take a shower
4) Get dressed
5) Eat breakfast
6) Carpool to work

What if steps 3 and 4 get switched?
Making Pancakes

- Put 2 eggs in a bowl.
- Add 1 glass of milk and mix.
- While the mixture doesn’t have the required consistency:
  - While the mixture is too liquid:
    - Add flour and mix.
  - While the mixture is too solid:
    - Add milk and mix.
- While the bowl is not empty:
  - Take a small cup of it.
  - Pour it in the frying pan and spread it.
  - When the back side is ready, turn it.
  - When the second side is ready, put it on the plate.
Driving: Turning Right

- Turn on the right signal.
- Take a look in the mirrors and by the right window.
- Slide to the right of your lane.
- Slow down.
- If the crossing lane is free:
  - Turn.
- Else:
  - While the crossing lane is not free:
    - Wait.
  - Turn.
Software Development Life Cycle, 6 parts

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

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

6) Maintenance
   ● Add, modify, and maintain the system
A Sample Problem:

Calculating the area and circumference of a circle

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

2) **Analysis:**

Determine the **input and output** of the program:

- Input: Radius of a circle
- Output: Area and Circumference of the circle

Determine the **formulas, fact, etc. Needed.**

- Area = PI * (Radius)^2
- Circumference = 2 * PI * Radius
- PI = 3.14159
3) 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) compute the Circumference
    (3.1) Circumference = 2 * PI * Radius
(4) print the Area and Circumference
4) Implementation:

→ your first VB program!
Solve the Sample Problem: the code!

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() 'read input for radius
    area = PI * radius * radius 'calculate area
    circumference = 2 * PI * radius 'calculate circumference

    'print out area and circumference
    Console.WriteLine("Area = " & area)
    Console.WriteLine("Circumference = " & circumference)

    Console.ReadLine() 'just to pause the program.

End Sub

End Module
Solve the Sample Problem: the code!

- The structure of the program
- Statements and comments
- Variables and constants
- Input and output
About Step 5: Testing and Debugging

- **Bug**
  - A mistake in a program

- **Debugging**
  - Eliminating mistakes in programs
  - Term used when a moth caused a failed relay on the Harvard Mark 1 computer. Grace Hopper and other programmers taped the moth in logbook stating: “First actual case of a bug being found.”
About Step 5: Testing and Debugging

- What kind of errors exist?
  - **Syntax errors**
    - Violation of the grammar rules of the language
    - Discovered by the compiler (compile-time error)
      - Error messages may not always show correct location of errors
  - **Run-time errors**
    - Error conditions detected by the computer at run-time i.e., while the program is executing
    - Have you seen the blue screen in windows?
  - **Logic errors**
    - Errors in the program’s algorithm
    - Most difficult to diagnose
    - *Computer does not recognize a logic error*
Questions (In Class Exercise)

- Can you…
  - List the steps in the software development life cycle?
  - Describe assembly code, machine code, high-level language code
  - Why using a high-level language is preferred for development
  - Describe the work of a compiler, linker, loader?
  - Define source code? Define object code?
  - Describe the three kinds of program errors?
  - What kind of error is produced if you type integr instead of integer?
  - Tell what type of error is produced when a program runs but produces incorrect results?