Problem Solving

Programming Process

- Programming is a process of problem solving
- Problem solving techniques
  - Analyze the problem
  - Outline the problem requirements
  - Design steps (algorithm) to solve the problem

Programming Process

This week

1. Clearly define what the program is to do.
2. Visualize the program running on the computer.
3. Use design tools such as a hierarchy chart, flowcharts, or pseudocode to create a model of the program.
4. Check the model for logical errors.
5. Type the code, save it, and compile it.
6. Correct any errors found during compilation. Repeat Steps 5 and 6 as many times as necessary.
7. Run the program with test data for input.
8. Correct any errors found while running the program. Repeat Steps 5 through 8 as many times as necessary.
9. Validate the results of the program.

Input, Processing, Output

Input, Processing & Output

Three steps that a program typically performs:

1) **Gather input data:**
   - from keyboard
   - from files on disk drives
2) **Process the input data**
3) **Display the results as output:**
   - send it to the screen
   - write to a file

Problem Solving

- The moment you read the problem, you should try to answer the following questions:
  - ***What information should the solution provide?***
  - ***What data do I have to work with?***
  - ***How to work with the data to deliver the needed solution?***
- The answer to the first question will tell you the problem **outputs; the answer to the second** question will tell you the problem **inputs; the answer to the third** question will tell you **the process.**

Example 1

- Develop a program to calculate area of a rectangle.
  1. Input data
     i) Length
     ii) width
  2. Process the input data
     i) Area=length*width
  3. Output data
     i) area

In-Class Exercise

- Do Lab 2, Exercise 3, No. 1, pg. 27
  – Identify the following information:
    1. Input data
    2. Process the input data
    3. Output data
- Do Lab 2, Exercise 3, No. 2, pg. 27
- Do Lab 2, Exercise 3, No. 3, pg. 28
- Do Lab 2, Exercise 3, No. 4, pg. 28

Representation of Algorithm

Problem Solving Methods in this Class

- 3 problem solving methods will be discussed in this class are:

1. Develop Algorithms
   - ❖ Flowchart
   - ❖ Pseudo code
2. Top-down design
   - ❖ Structured Chart

Algorithms

- Algorithm - a sequence of a finite number of steps arranged in a specific logical order to produce the solution for a problem.
- Algorithms requirements:
  i. Must have input
  ii. Must produce output
  iii. Unambiguous
  iv. Correctness
  v. Finiteness

Pseudo code

- Pseudocode is a semiformal, English-like language with limited vocabulary that can be used to design & describe algorithms.
- Purpose- to define the procedural logic of an algorithm in a simple, easy-to-understand for its readers.
- Free of syntactical complications of programming language.

Pseudo code

- Execution sequence follow the steps flow.

```
Example: Algorithm for
multiplying two numbers
1. Start
2. Get A
3. Get B
4. Calculate result
   C=A*B
5. Display result C
6. End
```

In-Class Exercise

- The following pseudocode algorithm has an error. Find the error.

```
area = width x length.
Display "What is the room's width?"
Input width.
Display "What is the room's length?"
Input length.
Display area.
```

In-Class Exercise

- Refer to your solution for :
  - Lab 2, Exercise 3, No. 1, pg. 27
  - Lab 2, Exercise 3, No. 2, pg. 27
- Develop a pseudo code for both questions.

In-Class Exercise

- Write pseudo code algorithms for the following.
  - A program that calculates a customer's available credit should ask the user for the following:
    - The customer's maximum amount of credit
    - The amount of credit used by the customer

    Once these items have been entered, the program should calculate and display the customer's available credit. You can calculate available credit by subtracting the amount of credit used from the maximum amount of credit.