LECTURE 2
(Sem 2, 17/18)

BITG 1233:
Programming &
Problem Solving

agvutem.wordpress.com
hafidzfazli.wordpress.com
Learning Outcomes

At the end of this lecture, you should be able to:

• appreciate programming as a tool to solve problems.
• explain the system (software or program) development methodology.
• describe the steps involved in problem solving.
• identify the input, the process and the output of a given problem.
• identify and apply basic problem solving design techniques to design algorithms in solving problems (by using flow chart and pseudo code).
Programming and Problem Solving

- What does problem solving have to do with programming? Well everything. Programming is about problem solving.

- Engineers and scientists are problem solvers. What a good problem solver does is often ask more questions. It's good to ask questions.
Solving a Problem

• When solving a problem, the level of abstraction that you choose to look at, can influence how "easy" or "hard" the problem is to solve. Solving a problem can involve moving between these levels of abstraction: more detailed, less detailed..

• You try the best you can, get an answer, and then try to improve. Always strive to improve.
Solving a Problem

• When ever you have to solve problems especially big, hard ones:

1. Don't give up.
2. Ask more questions.
3. Break the problem into smaller problems. (Breaking into parts, more like divide and conquer)
4. Think of the benefits and profit that you’ll get.
5. As soon as you develop one solution, start thinking of another way and just don't settle with what you’ve got until you can’t think of other solutions. This is most important quality of a problem solver.
Program Development (Programming)

• Programming is a creative process. A programming process critically determines the overall quality and success of the program. Therefore in a programming process, there is the outline of a plan (methodology) to follow.

• Most programming projects are built using a software development life cycle (SDLC) methodology. One of the popular models in SDLC is known as the waterfall model.
SDLC : The Waterfall Model

Software Requirements

Analysis

Design

Problem Solving Phase

Coding

Testing

Implementation Phase

Maintenance
Programming Process

• Involves TWO major Phases (with 4 steps of processes) that requires you to do the following in their order:

1. Problem Solving Phase
   - Involves Analysis (step 1) & Design (step 2)
   - The result is an algorithm that solves the problem.

2. Implementation Phase
   - Involves Coding (step 3)& Testing (step 4)
   - The result is a program that has been translated from the algorithm using a programming language.
Programming Approach

• To produce a program, we may use different programming approach.
• **Procedural programming** is a programming approach that focuses on the process.
• When procedural programming is used to solve big problems:
  a. It uses a top-down design to model the structure of the program.
  b. A big problem is divided into smaller sub problems.
  c. Solution of a sub problem actually involves a task or some tasks that are grouped as a module or sub-module, and are coded in a function of the program.
Step 1 - Analysis: To understand the problem.

- Clearly define what is needed for the **Input, Process & Output (IPO)** to solve the problem.

- If solution requires **top-down design**, then draw a **structure chart** that contains the hierarchy of modules in the solution.
Step 2 - Design the algorithm: To get the logical steps of achieving the problem’s solution.

- Algorithm is a set of well defined and precise instructions (or steps) that leads to a solution.
- **Algorithm design** - To determine how to take inputs and process them into the specified output.
- Algorithm must be tested for its correctness (no logic errors). Experience shows this saves time in getting your program to run correctly.
- Do corrections to the algorithm’s design as many times as necessary as long as there is still logic error in the algorithm.
• If any program is designed carefully using good and structured development techniques, the program will be efficient, error-free and easy to maintain.

• Algorithm can be designed using:
  a) Pseudo code or
  b) Flowchart
a) Pseudo code

• Precise algorithmic description of program logic.

• Its purpose is to describe, in precise algorithm detail, what the program being design is to do.

• Requires the defining the steps to accomplish the task in sufficient detail so that they can be converted into a computer program.
b) Flow Chart

- A schematic (diagram) representation of a sequence of operations in a process.

- Program flowcharts show the sequence of instructions in a single program or subroutine.

- Different symbols are used to represent different operations in a flowchart.
Basic Symbols of a Flow Chart Represent Various Operations

- **Direction**
- **Terminal** – beginning / end
- **Decision**
- **Connector**
- **Preparation**
- **Process**
- **Input / Output**
- **Predefined Process** (Call to a function)
Step 3 - Coding: Write, save and compile the program code using the IDE.

- Program is an algorithm that has been expressed in a programming language. The coding process is easier as you gain experience with the programming language.

- Correct any syntax errors found during program compilation.

- Do corrections to the program codes as many times as necessary until there is no more syntax error in the program.
Step 4 - Test the program: Run the program on sample data.

• Repeat Step 4 as many times as necessary to validate the result of the program.

• Correct any run-time and logic errors found. These two types of errors may require modification of the algorithm and program code, which means it need to repeat from step 2 or step 3.
Problem Solving Examples:

**Problem 1**

Problem Statement (or software requirements)

Your summer surveying job requires you to study some maps that give distances in miles. You and your co-workers prefer to deal in metric measurements (km). Write a program that performs the necessary conversion. The program will display the distance in km.

Formula: (1 mile = 1.609 kilometers)
a) **Solution of Problem 1 (using top down design)**

**Step 1: Analysis**

i. **Highlight the problem statement.**

Your summer surveying job requires you to study some maps that give distances in miles. You and your co-workers prefer to deal in **metric measurements** (km). Write a program that performs the **necessary conversion**. The program will display the distance in km. **Formula**: $1 \text{ mile} = 1.609 \text{ kilometers}$
ii. Identify: What are the INPUTs, PROCESSes, OUTPUTs?

• Input
  – the distance in miles: \( M \)

• Process (include the formula)
  – Convert distance in miles to kilometers
    (Formula given: 1 mile = 1.609 kilometers)
  – Therefore to calculate distance in kilometers:
    \( K = M \times 1.609 \)

• Output
  – the distance in kilometers: \( K \)
iii. Draw a Structure Chart (**only if problem solution uses top-down design)**

```
main
{conversion problem}

input
{distance in miles}

convert
{miles to kilometers}

display
{distance in kilometers}
```
Step 2: Design the Algorithm.

a) Algorithm using Pseudo code

1.0 Start
2.0 Call function input
   2.1 Read distance in miles : \( M \)
   2.2 Return M
3.0 Call function convert
   3.1 Calculate distance in kilometer : \( K = M \times 1.609 \)
   3.2 Return K
4.0 Call function display
   4.1 Display the distance in kilometers : \( K \)
5.0 Stop
b) Algorithm using Flow Chart

Start

M = input()

K = convert(M)

display(K)

Stop

input()

Read distance in miles (M)

Return M

convert (M)

K = M * 1.609

Return K

display (K)

Display distance in kilometers (K)

Return

LECTURE 2
/* Problem Solving Exercise : Introduction to C++ */
#include <iostream>
using namespace std;

double input(); //global declaration
void display(double K);
double convert(double M);

int main()
{
    double M, K; //local declaration
    //M represents distance in miles
    //K represents distance in kilometers

    M = input();
    K = convert(M);
    display(K);
    return 0;
}
double input()
{
    double miles;
    cout<<"Please enter distance in miles :";
    cin>>miles;
    return miles;
}

double convert(double distance_in_miles)
{
    double km;
    km = distance_in_miles * 1.609;
    return km;
}

void display(double distance_in_km)
{
    cout<< "Distance in kilometre is ";
    cout<< distance_in_km;
    cout<< "\n";
}
Step 4: Test the program.

Sample of the program run and its correct result (output):

```
Please enter distance in miles : 15
Distance in kilometre is 24.135
Press any key to continue . . . .
```
b) Solution of **PROBLEM 1** (without using top down design)

**Step 1: Analysis**

i. Highlight the problem statement.

Your summer surveying job requires you to study some maps that give distances in miles. You and your co-workers prefer to deal in metric measurements (km). Write a program that performs the necessary conversion. The program will display the distance in km.

Formula: \(1 \text{ mile} = 1.609 \text{ kilometers}\)
ii. Identify : What are the INPUTs, PROCESSes, OUTPUTs?

• Input
  – the distances in miles : *miles*

• Process (include the formula)
  – Calculate kilometers from miles.
    (Formula given : 1 mile = 1.609 kilometers)
    \[ kms = miles \times 1.609 \]

• Output
  – the distance in kilometers : *kms*
Step 2: Design the Algorithm.

**Flowchart**

**Pseudo code**

1.0 Start

2.0 Get the distance in miles: *miles*

3.0 Calculate the distance in kilometers: 
   
   \[ kms = miles \times 1.609 \]

4.0 Display the distance in kilometers: *kms*

5.0 Stop
Step 3: Code the program

```cpp
#include <iostream>
using namespace std;

#define mile_to_km 1.609

int main() {
    double miles, kms;

    cout << "Enter the distance in miles that need ";
    cout << "to be converted to kilometers : ";

    cin >> miles;

    kms = mile_to_km * miles;

    cout << endl << miles << " in miles is " << kms;
    cout << " in kilometers " << endl;
    return 0;
}```
Step 4: Test the program.

Sample of the program run and its correct result (output):

```
Enter the distance in miles that need to be converted to kilometers : 4.5
4.5 in miles is 7.2405 in kilometers
Press any key to continue . . .
```
**Problem 2**

Write a program that inputs 2 integers \( n1 \) and \( n2 \). Calculate and prints the value of \( n1^2 \) and \( n2^3 \).

**Solution of PROBLEM 2 (without using top down design)**

**Step 1: Analysis**

i. Highlight the problem statement.

Write a program that inputs 2 integers \( n1 \) and \( n2 \). Calculate and prints the value of \( n1^2 \) and \( n2^3 \).

ii. Identify:

- **Input** : \( n1, n2 \)
- **Process**
  - Calculate \( n1^2 \) :  \( ans1 = n1 \times n1 \)
  - Calculate \( n2^3 \) :  \( ans2 = n2 \times n2 \times n2 \)
- **Output** : \( ans1, ans2 \)
Step 2 : Design the algorithm

Pseudo code

1. Start
2. Input \( n1 \) and \( n2 \)
3. Calculate \( n1^2 \):
   \[ \text{ans1} = n1 \times n1 \]
4. Calculate \( n2^3 \):
   \[ \text{ans2} = n2 \times n2 \times n2 \]
5. Print \( \text{ans1}, \text{ans2} \)
6. Stop
Step 3: Code the program

#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    double n1, n2, ans1, ans2;
    cin >> n1 >> n2;
    ans1 = pow(n1, 2);
    ans2 = pow(n2, 3);
    cout << ans1 << endl;
    cout << ans2 << endl;
    return 0;
}

Step 4: Test the program

Sample of the program run and its correct result (output):
EXAMPLE OF **PROBLEM 3** (without using top down design)

**Step 1: Analysis**

i. **Highlight the problem statement.**

Ohm's law is often expressed in the form \( V = IR \), where \( V \) is the voltage measured in volts, \( I \) is the current measured in amps, and \( R \) is the resistance measured in ohms.

Write a program that can calculate how much current (\( I \)) would flow through an input ohm of resistor (\( R \)) if you were to connect the resistor to an input volts of power supply (\( V \)).

However you need to check for the input volts of the power supply. If it is > 10 volts, don’t calculate the current, you just print out a message “The volts is too big”. If the volt is <= 10, calculate and print the current (\( I \)) and if the volts is negative, print out a message “Not a valid input”.
ii. Identify:

- Input: $V, R$
- Process (include the Formula)
  - If $V > 10$, *print message “The volts is too big”*
  - If $V \leq 10$, $I = V/R$
  - If $V < 0$, *print message “Not a valid input”*
- Output: $I$
Step 2: Design the algorithm

Pseudo code

1.0  Start

2.0 Get the input data for V, R

3.0 If V > 10
    print message “The volts is too big”
else
    If V<0 , print msg “Not a valid input”
    else
        If V<=10, I = V/R, print I

4.0  Stop
#include <iostream>
using namespace std;

int main( )
{
    double V,R,I;
    cout << "Enter V : "; cin >> V ;
    cout << "Enter R : "; cin >> R;
    if (V > 10)
        cout << "The volts is too big\n";
    else
        if (V < 0)
            cout << "The input is invalid\n";
        else
            if (V < 10)
            {
                I = V/R;
                cout << "I = " << I << endl;
            }
    return 0;
}
Step 4: Test the program.

Sample of the program run and its correct result (output):

![Program output image](image-url)
Sample 2 of the program run and its result (output):

Enter V : 15
Enter R : 7
The volts is too big
Press any key to continue . . .
Sample 3 of the program run and its result (output):

Enter U : -8
Enter R : 3
The input is invalid
Press any key to continue . . .
Can you...

– Describe the first step to take when creating a program?

– List the two main phases of the programming process?

– Explain the importance of the problem-solving phase?

– List the steps in the software development life cycle?
Identify Input, Process and Output to solves each problem. Then write pseudo code and draw the flowchart for each one.

Problem 1

Calculate and display the area of a circle based on a given radius.

Formula: \text{Area of a circle} = 3.14 \times \text{radius} \times \text{radius}

Problem 2

In a math subject Sam has taken 5 monthly tests. Input the marks Sam has secured in each of the tests and find out his average mark. Then display the result.

Problem 3

Solve the problem using top down design approach to calculate the sum, average, difference, product, quotient and remainder of two integer numbers given by the user. Use the main function to display their results.

-\text{THE END}-