Fundamentals of Programming
STUDY GUIDE FOR MODULE NO. 1
INTRODUCTORY CONCEPTS TO PROGRAMMING

MODULE OVERVIEW

This module aims to enable you to gain and demonstrate knowledge of computer systems, programming tools and basic steps in problem solving as preparation to understanding programming. The last section of this module provides you with an example on how to translate an algorithm design into Java program codes. In addition, basic debugging technique is illustrated. Short discussions and self-tests were included to enrich learning of the student. Finally, an activity will have to be accomplished to assess your proficiency.

LEARNING CONTENTS (Java Programming Tools)

The very first thing you need to do programming is a compiler. A compiler reads the entire program and converts it into object code, which is a translation of the program source code into a form that the computer can execute directly. There are a number of commercial Integrated Development Environments (IDEs) that support Java. There are also some good free IDEs available, including Eclipse . These are completely self-contained environment for creating, compiling, linking and testing Java programs. It incorporates a range of fully integrated tools designed to make the whole process of writing Java programs easy.

LEARNING CONTENTS (Basic Steps in Problem Solving)

Introduction

Problem solving is the act of defining a problem; analysing the problem; selecting alternatives for a solution; and implementing a solution. In the context of programming, basic steps in problem solving are problem definition, problem analysis and algorithm design. In this unit, you will be familiarized in these basic steps in preparation for program codes writing in C++.

Unit learning outcomes

By the end of this unit you should be able to:

- Analyse a given problem by identifying input, process and output.
- Familiarize with keywords and symbols used in pseudocode.
- Formulate algorithms and represent them using pseudocode.

1.3 Basic Steps in Problem Solving

❖ Problem solving for programming consists of three steps: problem definition, problem analysis and algorithm design.

1. Defining and Analyzing the Problem

A. Stating the Problem

Problem 1:

Write a program to determine the sum of two numbers.

Problem 2:

Write a program that will compute the area of a circle.

B. Definitive Statement of the Problem

Problem 1:

Write a program to determine the sum of two numbers. Give the computer two numbers. It should be able to add the two numbers and give the result.

Problem 2:

Write a program that will compute the area of a circle. How will you re-state Problem No 2?

C. Analyzing the Problem NATURE. (What is it about) SCOPE. (Limitations)

Example 1

Problem 1:

Write a program to determine the sum of two numbers.
Nature: Mathematical
Scope: Limit the scope to numbers that are integers or whole numbers.

Example 2

Problem 2

Write a program that will compute the area of a circle.
Nature: Geometry
Scope: Allow fractions and decimals as acceptable values. No negative values for the radius should be accepted.

C.1. Identifying the Output

Problem 1: Write a program to determine the sum of two numbers. **Output:**

✓ [sum]
✓ The sum of the two numbers you gave is: [sum]

C.1. Identifying the Input

Problem 1: Write a program to determine the sum of two numbers.

Input:

✓ We need two numbers as the input. (keyed in the keyboard)

C.1. Identifying the Input

Problem 2:

Write a program that will compute the area of a circle.

What is the Input Required?

D. Defining the Process

Problem 1:

Write a program to determine the sum of two numbers.

Process:

✓ Enter two numbers using the keyboard. The computer must be able to reject fractions and decimal numbers. The computer will then add the two numbers. The result of the addition will be displayed on the computer screen.

Problem 2:

Write a program that will compute the area of a circle.

Process:

✓ The program must be able to accept a number representing the radius. The computer should make it sure that no negative number will be entered. The computer must compute the area of the circle. Result must be displayed on the screen.

2. Planning the Solution

Algorithm

- o A set of rules for solving a problem in a finite number of steps.

Overview of an Algorithm

I. STEPS IN COOKING SCRAMBLED EGG

- Prepare all necessary items needed (egg, pan, oil etc.)
- Beat the egg
- Add salt o Heat pan
- Put oil into the pan
- Pour beaten egg if pan is already hot enough
- Cook egg for at least two minutes o Serve Egg

Control Structures

Structured programming

❖ is a technique that emphasizes the breaking of a program or algorithm into logical sections, or modules following a universal programming standard. (Breaking of programs into smaller modules)

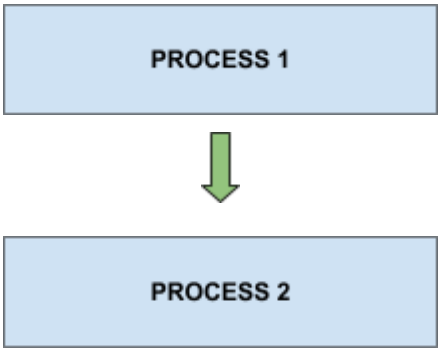
❖ Control structures is a step, or a series of steps, that determines what instructions or set of instructions will be done next.

Three Basic Structures

➢ Sequential
➢ Selection
➢ Iteration

Sequential

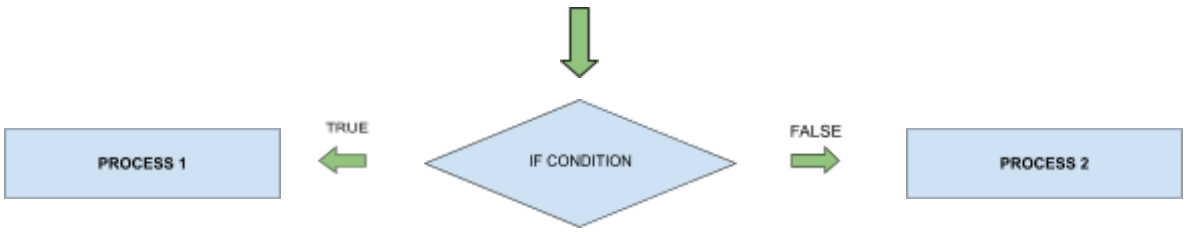
❖ is the most straight forward. The program will just follow the instructions one after the other in a sequential manner.



Selection

❖ Let your program or algorithm make a decision. Choose one from two sets of instructions depending on the condition.

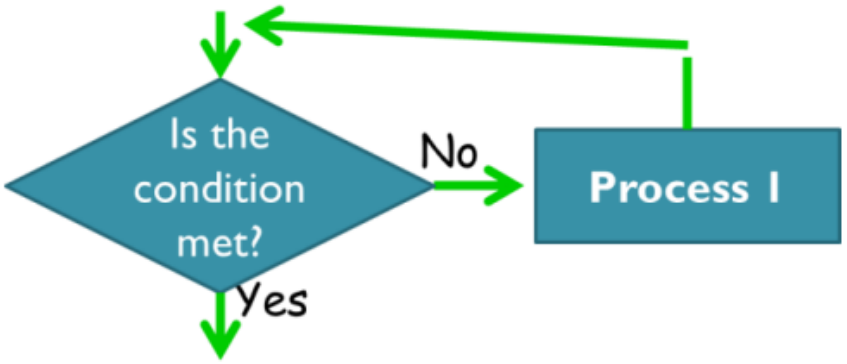
- IF (condition) THEN (Process 1) ELSE (Process 2)



Iteration

❖ We use when we want our algorithm or program to repeat a process a specified number of times.

❖ The structure that repeats an instruction or a set of instructions until a condition is met is known as iteration or loop.



Flowcharting

❖ is one method of pictorially representing a procedural (step-by-step) solution to a problem before you actually start to write the computer instructions required to produce the desired results.

❖ Two Types of Flowchart

o System (data) flowcharts

▪ Defines the major phases of the processing, as well as the various data media used.

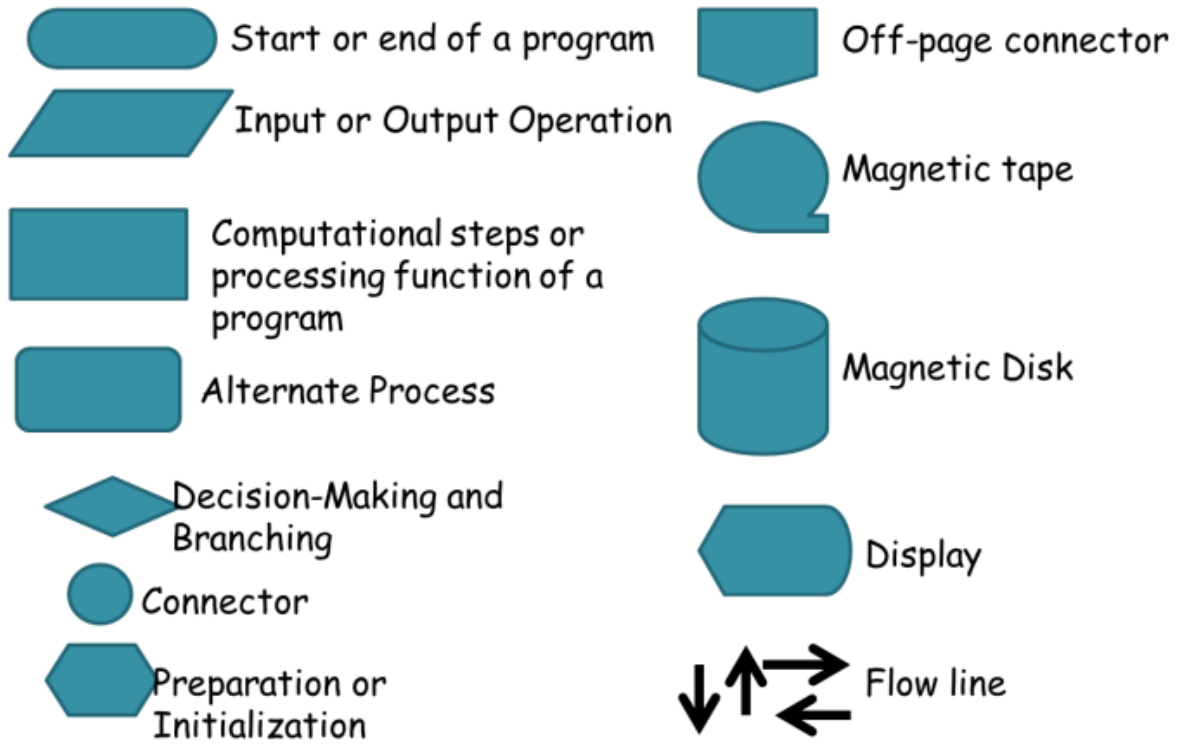
o Programming flowcharts

Example of system flowchart

Programming Flowchart

❖ Constructed by the programmer to represent the sequence of operations the computer is to perform to solve a specific problem.
❖ It graphically describes what is to take place in the program

Standard Symbol

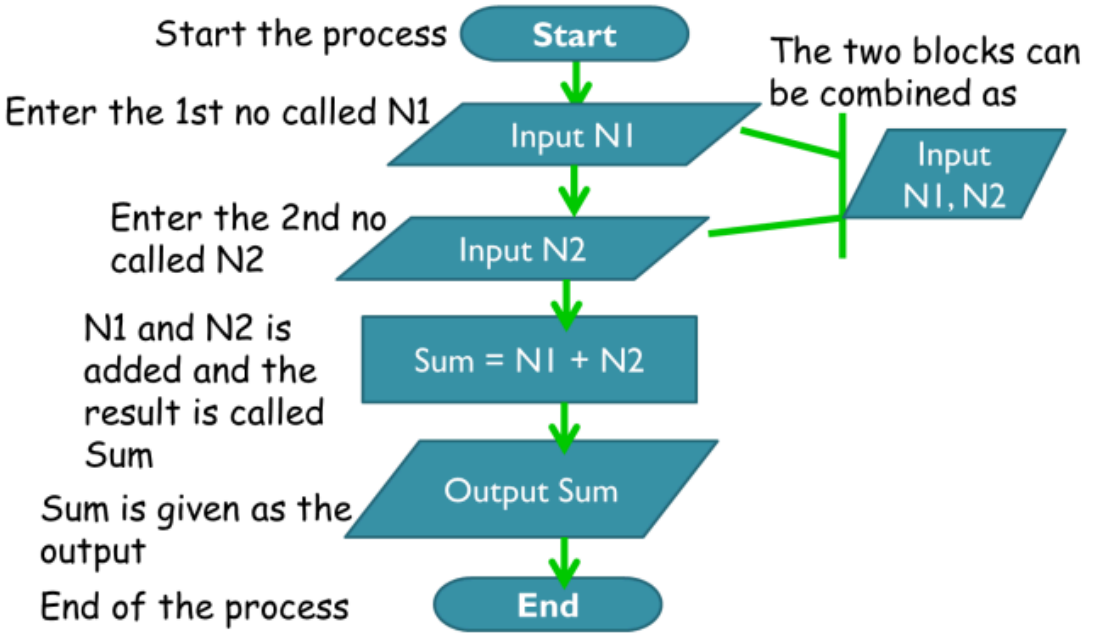


Examples on Flowcharting

Problem: Make a flowchart to determine the sum of two numbers.
Definitive Statement: Give the computer two numbers. It should be able to add the two numbers and give the result.

Problem: Make a flowchart to determine the sum of two numbers.
Definitive Statement: Give the computer two numbers. It should be able to add the two numbers and give the result.

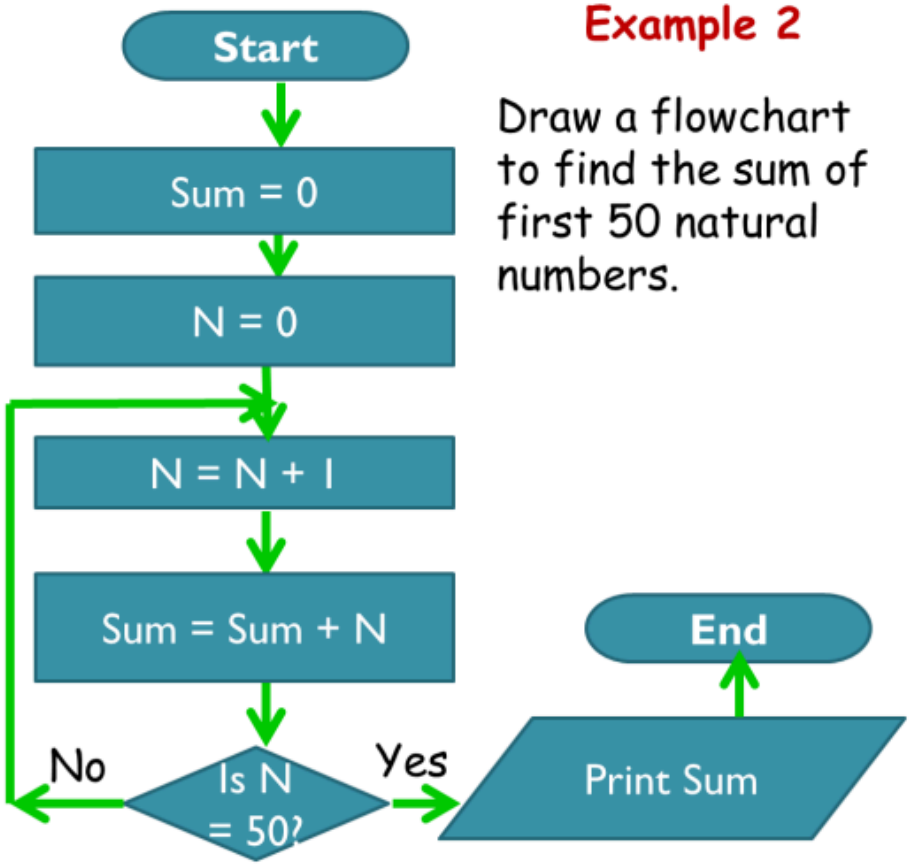
Output: [sum] or The sum of the two numbers is: [sum]
Input: we need two numbers as the input. The program allows both positive and negative integers.

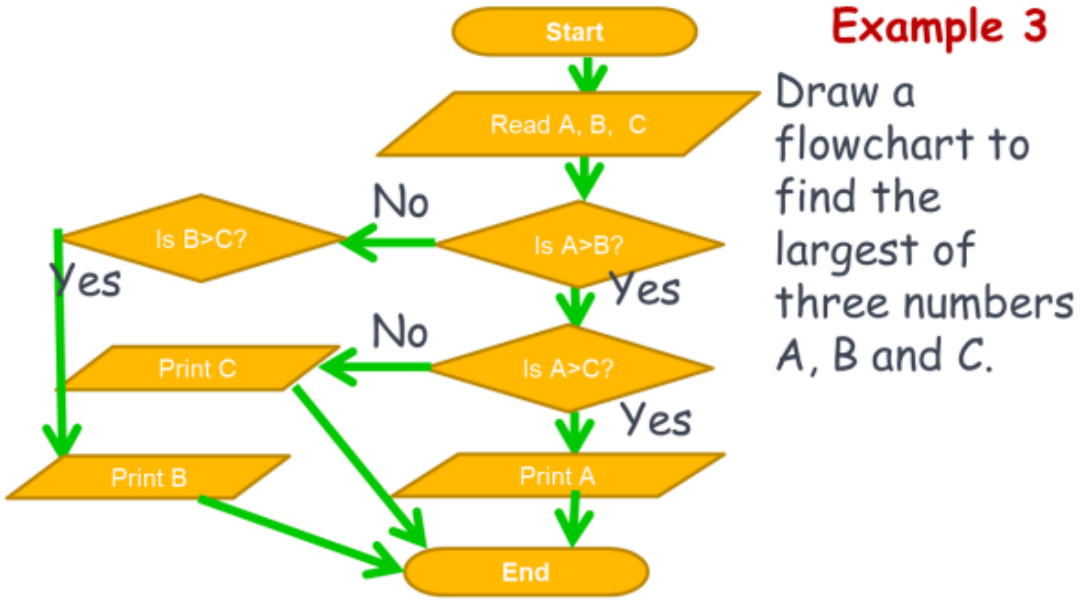


Example 1

Example 2

Draw a flowchart to find the sum of first 50 natural numbers.





Pseudocode

❖ Uses words and mathematical expressions to represent the logic, or the flow of the program.

❖ Informal language for writing algorithms.

Example of Pseudocode

Problem: Write a pseudocode to determine the sum of two numbers.

Definitive Statement: Give the computer two numbers. It should be able to add the two numbers and give the result.

Output: The output of the computer should give us a number which is integer or a whole number. [sum] or The sum of the two numbers you gave is: [sum]

NOTATION	USE	OTHER NOTATIONS USED
INPUT	Used when reading or entering data in any of the input devices.	READ, ENTER
OUTPUT	Used when we wish a value to be displayed.	WRITE, DISPLAY, PRINT
BEGIN	Marks the start of a pseudocode.	START
END	Marks the end of a pseudocode.	STOP, HALT
←	Storage Operator. Used to indicate that a value is stored in a memory location.	

Example of Pseudocode

Problem: Write a pseudocode to determine the sum of two numbers.

Definitive Statement: Give the computer two numbers. It should be able to add the two numbers and give the result.

Output: The output of the computer should give us a number which is integer or a whole number. [sum] or The sum of the two numbers you gave is: [sum]

Pseudocode:

BEGIN INPUT the first number
INPUT the seocnd number
Add the two numbers
OUTPUT the sum
END

LEARNING CONTENTS (Coding and Debugging)

Introduction

Now that you have learned the basic steps in problem solving and creating algorithms using pseudocode or flowchart, you are now ready to transform your algorithm into Java program codes. This process is called coding. After learning how to use Integrated Development Environment (IDE), you are now ready to write and run your Java program using Eclipes. As soon as you start coding, it is normal for beginners to encounter instances wherein your program would not execute. When this happens, error messages are displayed. The only way you can execute your program is to find the errors or otherwise termed bugs and fix them. We call this process as debugging – which means finding the bugs and correcting them.

Unit learning outcomes

By the end of this unit you should be able to:

- Transform an algorithm into Java program codes.
- Use IDEs in running and debugging programs.

1.4 Coding and Debugging

The best way to learn programming language is by writing programs. Although typically the first program a beginner would write is “Hello World”, a good way to start is write your algorithm using any representation such as pseudocode or flowchart and then later transform each part into equivalent Java codes. Table below illustrates lines of a pseudocode translated into equivalent Java codes.

Pseudocode lines	Equivalent Codes in Java
BEGIN	{
INPUT A	int A = myObj.nextInt()
INPUT B	int B = myObj.nextInt()
Sum ← A + B	Sum = A+B;
Ave ← Sum /2	Ave = Sum / 2;
Output Ave	System.out.println(Ave);
END	}

The Scanner class is used to get user input, and it is found in the java.util package.To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation. In our example, we will use the nextLine() method, which is used to read Strings:

However, when you start coding using Eclipse you need to complete your program with Class Main, enclose the lines of code previously shown, inside public static void (String[] args) just before the close curly brace (}). In addition, variables used must be declared before use. If you are not familiar with these terms, do not worry because these will all be discussed to you in detail as you proceed to the other modules. Below is an illustration of how you do coding in Eclipse. Once you have written your code, you can try running your program by going to Build and selecting Build and run. If all is well, an output window displaying results will show. Otherwise, build errors will be displayed.

```java
import java.util.Scanner;

public class AverageCalculator {
    public static void main(String[] args) {
        // Create a Scanner object to read input from the user
        Scanner scanner = new Scanner(System.in);

        // Input A
        System.out.print("Enter the first number (A): ");
        int A = scanner.nextInt();

        // Input B
        System.out.print("Enter the second number (B): ");
        int B = scanner.nextInt();

        // Calculate the sum
        int sum = A + B;

        // Calculate the average
       int ave = sum / 2;

        // Output the average
        System.out.println("The average is: " + ave);

        // Close the scanner
        scanner.close();
    }
}
```

Output

```
Enter the first number (A): 10
Enter the second number (B): 20
The average is: 15.0
```