CCS20

Fundamentals of Programming

Module 1 – Introduction to Programming

ANGELES UNIVERSITY FOUNDATION
COLLEGE OF COMPUTER STUDIES
BSIT

CCS20
Fundamentals of Programming

Agenda Style

01

Introduction

What is Programming?

02

Algorithm

Understand the process using Algorithm

03

Flowchart

Understand and apply flowchart based on the given algorithm

04

Pseudocode

What is Pseudocode?

Computer Program

- A **computer program** (also a **software program**, or just a **program**) is a sequence of instructions written to perform a specified task for a computer. A computer requires programs to function, typically executing the program's instructions in a central processor. The program has an executable form that the computer can use directly to execute the instructions.

Source Code

- **Source code** is text written in a computer programming language. Such a language is specially designed to facilitate the work of computer programmers, who specify the actions to be performed by a computer mostly by writing source code, which can then be automatically translated to binary machine code that the computer can directly read and execute.

Computer Programming

- **Computer programming** (often shortened to **programmin**g or **coding**) is the process of designing, writing, testing, debugging/ troubleshooting, and maintaining the source code of computer programs.
- This source code is written in a programming language.
- The purpose of programming is to create a program that exhibits a certain desired behavior.
- The process of writing source code often requires expertise in many different subjects, including knowledge of the application domain, specialized algorithms and formal logic.

Introduction to Programming

- **Program:** self-contained set of instructions used to operate a computer to produce a specific result
    - Also called **software**
- **Programming:** the process of writing a program, or software

Machine Language

- **Machine language programs,** also called **executables,** consist of binary instructions
- Each instruction has two parts:
    - **Instruction** part: the operation to be performed; also called an **opcode**
    - **Address** part: memory address of the data to be used
- Each class of computer has its own particular machine language
- Writing in machine language is tedious!

Assembly Languages

- **Assembly Language:** programming language with symbolic names for opcodes, and decimals or labels for memory addresses
  Example:

```
ADD  1, 2

MUL  2, 3
```

- Assembly language programs must be translated into machine instructions, using an **assembler**

Introduction to Programming:

- **Low-level languages:** languages that use instructions tied directly to one type of computer
  Examples: machine language, assembly language

- **High-level languages**: instructions resemble written languages, such as English,and can be run on a variety of computer types
  Examples: Visual Basic, C, C++, Java

Introduction to Programming: (continued)

- **Source code:** the programs written in a high- or low-level language
- Source code must be translated to machine instructions in one of two ways:
    - **Interpreter:** each statement is translated individually and executed immediately after translation
    - **Compiler:** all statements are translated and stored as an executable program, or **object program;** execution occurs later

Low- and High-Level Languages (continued)

- Large C++ programs may be stored in two or more separate program files due to
  - Use of previously written code
  - Use of code provided by the compiler
  - Modular design of the program (for reusability of components)
- **Linker**: combines all of the compiled code required for the program

Low- and High-Level Languages (continued)

- Programs can also be classified by their orientation:
  - **Procedural:** available instructions are used to create self-contained units called procedures
  - **Object-oriented:** reusable objects, containing code and data, are manipulated
- Object-oriented languages support reusing existing code more easily

Program Development Cycle

1. Analyze: Define the problem
2. Design: Plan the solution to the problem
3. Choose the Interface: Select the objects
4. Code: Translate the algorithm into a programming language.
5. Debug and Test: Locate and remove any errors in the program.
6. Complete the Documentation: Organize all the materials that describe the program.

Algorithms

- **Algorithm:** the step-by-step sequence of instructions that describe how the data is to be processed to produce the desired output
- Programming = the translation of the selected algorithm into a language the computer can use

Algorithms (continued)

- **Pseudocode:** English-like phrases used to describe the algorithm
- **Formula:** description of a mathematical equation
- **Flowchart:** diagram showing the flow of instructions in an algorithm
  - Flowcharts use special symbols

Expressing Algorithms

- Natural Languages
- Pseudocode
- Flowcharts
- Programming Languages

Pseudocode

- A program design technique that uses English words.
- Has no formal syntactical rules.
- ***Pseudo*** means false, thus pseudocode means false code. It looks like (imitates) real code but it is NOT real code.
- Pseudocode cannot be compiled nor executed, and there are no real formatting or syntax rules.
- Pseudocode should not include keywords in any specific computer languages.
- The benefit of pseudocode is that it enables the programmer to concentrate on the algorithms without worrying about all the syntactic details of a particular programming language.

How do I write Pseudocode?

- First you may want to make a list of the main tasks that must be accomplished on a piece of scratch paper.
- Then, focus on each of those tasks. Generally, you should try to break each main task down into very small tasks that can each be explained with a short phrase.
- There may eventually be a one-to-one correlation between the lines of pseudocode and the lines of the code that you write after you have finished pseudocoding.

- It is not necessary in pseudocode to mention the need to declare variables. It is wise however to show the initialization of variables.

- **All statements showing "dependency" are to be indented. These include while, do, for, if, switch.**

Some Keywords That Should be Used

For looping and selection, the keywords that are to be used include:

Do While...EndDo	Call ... with (parameters)
Do Until...Enddo	Call
Case...EndCase	Return
If...Endif	When

As verbs, use the words:

Generate, Compute, Process
Set, Reset, Increment, Compute, Test
Calculate, Add, Sum, Multiply, ...
Print, Display, Input, Output, Edit

Pseudocode Examples

Original Program Specification:

Write a program that obtains two integer numbers from the user. It will print out the sum of those numbers.

Pseudocode:

Prompt the user to enter the first integer
Prompt the user to enter a second integer
Compute the sum of the two user inputs
Display an output prompt that explains the answer as the sum
Display the result

Original Program Specification:

Write a program that will display if the entered student's grade is passed of failed. Passing grade is 70.

Pseudocode:

```
Prompt the user to enter a student grade
If student's grade is greater than or equal to 70
        Print "passed"
else
        Print "failed"
```

Original Program Specification:

Write a program that will allow the student to input the grade on his 5 subjects and will compute for his average grade.

Pseudocode example : If then Else

```
If age > 17
        Display a message indicating you can vote.
Else
        Display a message indicating you can't vote.
Endif
```

Pseudocode example : Case

```
Case of age
        0 to 17   Display "You can't vote."
       18 to 64   Display "Your in your working years."
       65 +       Display "You should be retired."
Endcase
```

Pseudocode example : While loop

```
count assigned zero
While count < 5
        Display "I love computers!"
         Increment count
Endwhile
```

Pseudocode example : For loop

```
For x starts at 0, x < 5, increment x
          Display "Are we having fun?"
Endfor
```

Pseudocode example : Do While loop

```
count assigned five
Do
        Display "Blast off is soon!"
        Decrement count
While count > zero
```

Pseudocode example : Repeat Until loop

```
count assigned five
Repeat
        Display "Blast off is soon!"
        Decrement count
Until count < one
```

Pseudocode example : Function with no parameter passing

```
Function clear monitor
        Pass In: nothing
        Direct the operating system to clear the monitor
                Pass Out: nothing
Endfunction
```

Pseudocode example : Function with parameter passing

```
Function delay program so you can see the monitor
        Pass In: integer representing tenths of a
second                  Using the operating system delay
the program
        Pass Out: nothing
Endfunction
```

Pseudocode of : Function main calling the clear monitor function

```
Function main
        Pass In: nothing
        Doing some lines of code
        Call: clear monitor
        Doing some lines of code
        Pass Out: value zero to the operating system
Endfunction
```

Flowcharts Symbols

	Terminator (Terminal Point, Oval)	Terminators show the start and stop points in a process. When used as a Start symbol, terminators depict a *trigger action* that sets the process flow into motion.
	Line (Arrow, Connector)	Flow line connectors show the direction that the process flows.
	Decision	Indicates a question or branch in the process flow. Typically, a Decision flowchart shape is used when there are 2 options (Yes/No, No/No-Go, etc.)
	Data (I/O)	The Data flowchart shape indicates inputs to and outputs from a process. As such, the shape is more often referred to as an I/O shape than a Data shape.
	Process	Show a Process or action step. This is the most common symbol in both process flowcharts and business process maps.

Flowcharts Symbols (continued)

Connector (Inspection)

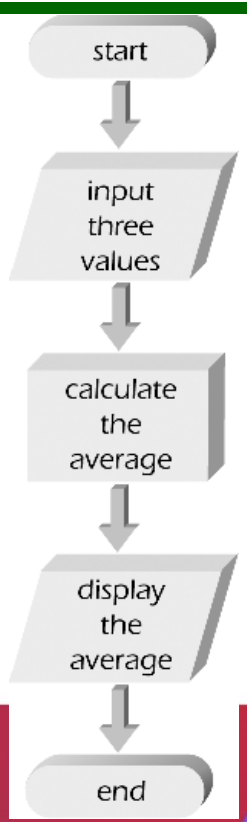
Flowchart: In flowcharts, this symbol is typically small and is used as a Connector to show a jump from one point in the process flow to another. Connectors are usually labeled with capital letters (A, B, AA) to show matching jump points. They are handy for avoiding flow lines that cross other shapes and flow lines. They are also handy for jumping to and from a sub-processes defined in a separate area than

Off-Page Connector

Off-Page Connector shows continuation of a process flowchart onto another page. When using them in conjunction with Connectors, it's best to differentiate the labels, e.g. use numbers for Off-Page Connectors and capital letters for Connectors. In actual practice, most flowcharts just use the Connect shape for both on-page and off-page references.

Flowcharts (continued)

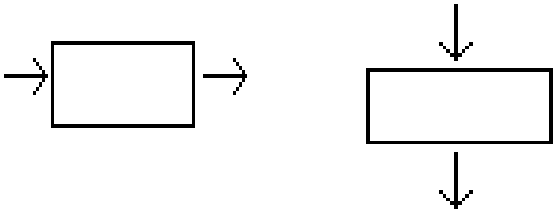
Flowchart for calculating the average of three numbers.



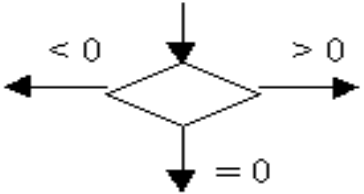
Guidelines for Drawing a Flowchart

- In drawing a proper flowchart, all necessary requirements should be listed out in logical order.
- The flowchart should be clear, neat and easy to follow. There should not be any room for ambiguity in understanding the flowchart.
- The usual direction of the flow of a procedure or system is from left to right or top to bottom.

- Only one flow line should come out from a process symbol.



- Only one flow line should enter a decision symbol, but two or three flow lines, one for each possible answer, should leave the decision symbol.



- If the flowchart becomes complex, it is better to use connector symbols to reduce the number of flow lines. Avoid the intersection of flow lines if you want to make it more effective and better way of communication.

- Ensure that the flowchart has a logical *start* and *finish.*

- It is useful to test the validity of the flowchart by passing through it with a simple test data.

Advantages of using flowcharts

The benefits of flowcharts are as follows:

- Communication: Flowcharts are better way of communicating the logic of a system to all concerned.

- Effective analysis: With the help of flowchart, problem can be analyzed in more effective way.

- Proper documentation: Program flowcharts serve as a good program documentation, which is needed for various purposes.

- Efficient Coding: The flowcharts act as a guide or blueprint during the systems analysis and program development phase.
- Proper Debugging: The flowchart helps in debugging process.
- Efficient Program Maintenance: The maintenance of operating program becomes easy with the help of flowchart. It helps the programmer to put efforts more efficiently on that part

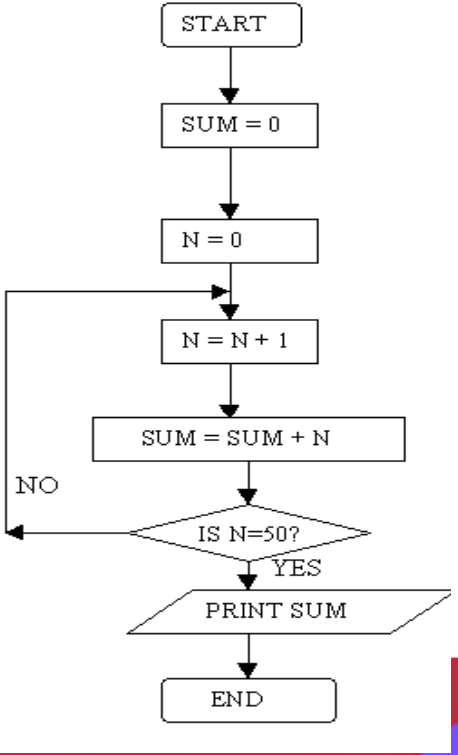
Limitations of using flowcharts

- Complex logic: Sometimes, the program logic is quite complicated. In that case, flowchart becomes complex and clumsy.
- Alterations and Modifications: If alterations are required the flowchart may require re-drawing completely.
- Reproduction: As the flowchart symbols cannot be typed, reproduction of flowchart becomes a problem.

Flowcharting Examples

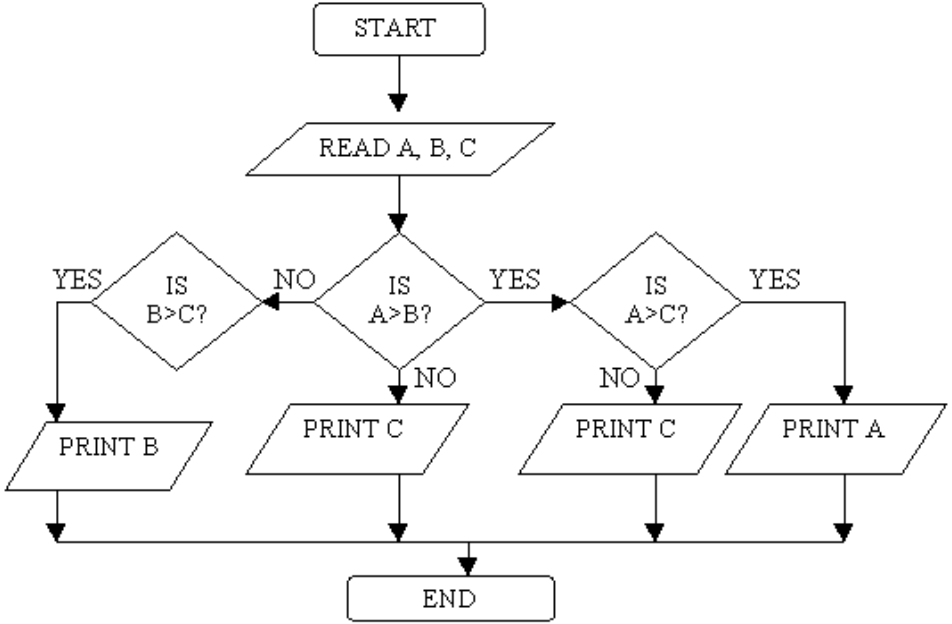
Example 1

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



Example 2

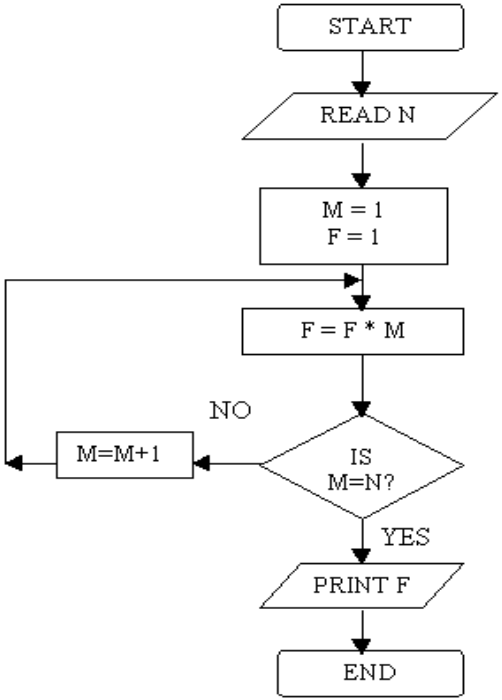
Draw a flowchart to find the largest of three numbers A,B, and C.



Example 3

Draw a flowchart for computing factorial N (N!)

Where N! = 1 ´ 2 ´ 3 ´ …… N .



Problem Solving Process

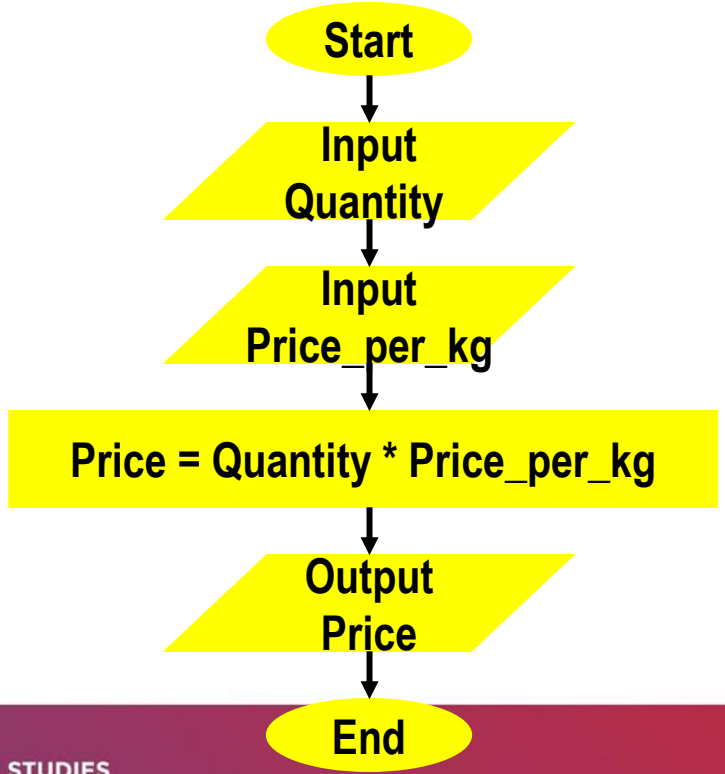
Example Problem #1

Calculate and display the price of a number of apples if the quantity in kg and price per kg are given.

Pseudocode

1. Start
2. Read **quantity**
3. Read **price_per_kg**
4. **price** ← **quantity** * **price_per_kg**
5. Print **price**
6. End

Flowchart: Calculate Price of Apples



Example #2

A car park has the following charges:

The 1st hour costs Php 5.00. The subsequent hour cost Php 2.00 per hour. Write an algorithm based on a vehicle's entry and exit time.

Logic Formulation

The VP Academic Affairs ask you to develop a program that will compute the students grade using the grading system. It will also determine if the student will get a passing grade. The user/teacher should input the following grades and it will automatically display the student name, computed result, as well as the remarks.

Grading System = Average of PG, MG, FG.
Passing Grade is 75 and above.