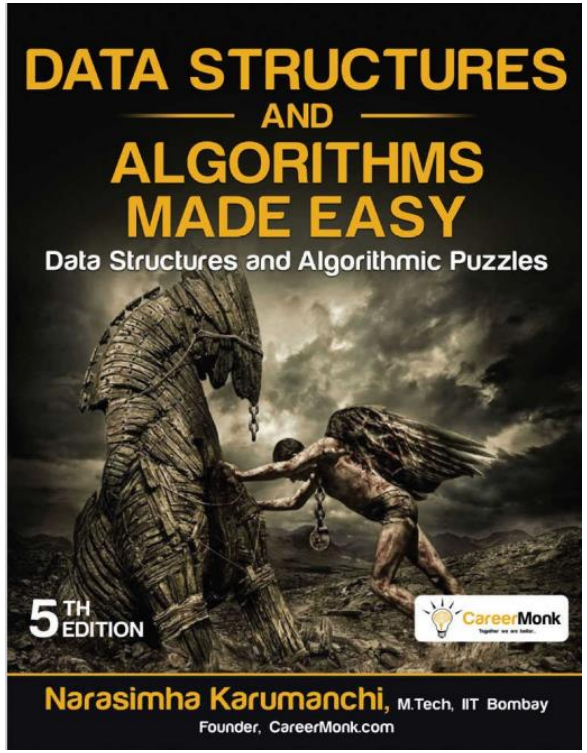
Programming



Chapter 1

Introduction to algorithm

BTEC Alliance with FPT Education

Pearson

Course introduction

- This unit introduces students to the core concepts of programming with an introduction to algorithms and the characteristics of programming paradigms
- On successful completion of this unit students will be able to design and implement a simple computer program in a chosen language (C#) within a suitable IDE (Visual Studio .NET)

Learning outcomes

- LO1: Define basic algorithms to carry out an operation and outline the process of programming an application.
- LO2: Explain the characteristics of procedural, object-orientated and event-driven programming, conduct an analysis of a suitable Integrated Development Environment (IDE)
- LO3: Implement basic algorithms in code using an IDE.
- LO4: Determine the debugging process and explain the importance of a coding standard

Course materials

- Login to CMS (http://cms.btec.edu.vn/) with student account (name@fpt.edu.vn)
- Choose correct class
- Enroll by class name (GCH0x0x)
- CMS Folder:
  - Slides
  - Demo
    - Assignment 1
    - Assignment 2

Course preparation

- Drawing tools (choose one):
  - Visio
  - Draw.io or Lucichart (online)
  - Astah (recommendation, using student email to register full version)
- IDE
  - Visual Studio Community 2022

Course reading

- Slides on CMS
- C# 6.0 and .NET Framework 4.6
  - http://library.books24x7.com/toc.aspx?bookid=104367
  - Registered with student email
- Mohammad Rahman, Expert C#5.0 with .NET 4.5 Framework

Problem solving

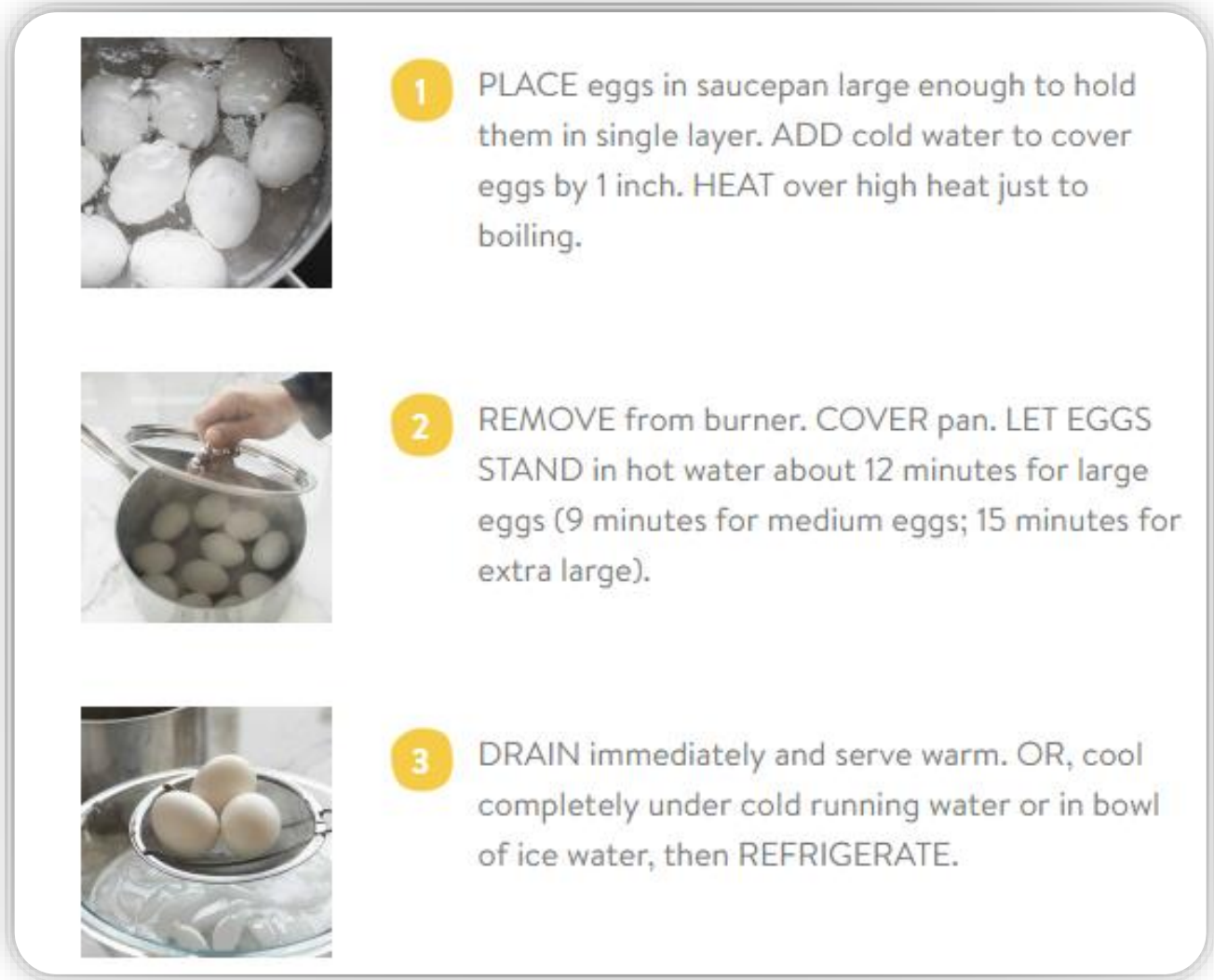
- It’s a creative process, it is an act of
  - Defining a problem
  - Determining the cause of the problem
  - Identifying, prioritizing, and selecting a
  - Implementing a solution

BTEC Alliance with FPT Education

Pearson

Algorithm

- An algorithm is the step-by-step unambiguous instructions to solve a given problem.
- In the traditional study of algorithms, there are two main criteria for judging the merits of algorithms: correctness (does the algorithm give solution to the problem in a finite number of steps?) and efficiency (how much resources (in terms of memory and time) does it take to execute.



http://www. incredibleegg.org

Computers and Algorithms

- To make a computer do anything, we have to write a computer program.
- To write a computer program, we have to tell the computer, step by step, what we want it to do.
- The computer then "executes" the program, following each step mechanically, **to accomplish the desired goal**.
- **Data structure + algorithms = program**

Computers and Algorithms

- When we are telling the computer what to do, we also get to choose how it's going to do it.
- That's where computer Algorithms come in to picture.
- The algorithm is the basic technique used to get the job/work done.
- When we say that algorithms are ubiquitous, we mean… Algorithms are found everywhere. Algorithms can be used to solve many different types of problems. Algorithms are a daily occurrence.

Algorithms Characteristics

- Finiteness: An algorithm must always terminate after a finite number of steps.
- Definiteness: Each step of an algorithm must be precisely defined; the actions to be carried out must be rigorously and unambiguously specified for each case.
- Input: An algorithm has zero or more inputs
- Output: An algorithm has one or more outputs
- Effectiveness: the algorithms is based on the available resources. It should be effective, exact and suitable for programs.
- Language Independent: The Algorithm designed must be language-independent. It has to be general resources. We can apply to all programming languages.

Types of algorithms

- Sequence: this type of algorithm is characterized with a series of steps, and each step will be executed one after another.
- Branching: this type of algorithm is represented by the "if-then" problems. If a condition is true, the output will be A, if the condition is false, the output will be B.
  This algorithm type is also known as "selection type".
- Loop: for this type, the process might be repeatedly executed under a certain condition. It is represented by "while" and "for" problems. But make sure the process will end after a number of loops under the condition. This algorithm type is also known as "repetition type".

Algorithm representation

- Natural language
- Flowchart
- Pseudo code

Natural language

- List sequentially the steps in natural language to represent the algorithm
- Advantages: Simple, no knowledge of representation (pseudocode, flowchart,...)
- Disadvangtages: Long, unstructured. Sometimes it's hard to understand, can't express the algorithm

Flowchart

- Flowchart is a diagrammatic representation of an algorithm
- It uses different symbols to represent the sequence of operations, required to solve a problem
- It serves as a blueprint or a logical diagram of the solution to a problem

Flowchart

- The first design of flowchart goes back to 1945 which was designed by John Von Neumann.
- Unlike algorithm, Flowchart uses different symbol to design a solution to a problem
- It is another commonly used programming tool. By looking at a flowchart one can understand the operations and sequence of operations performed in a system.
- Flowchart is often considered as a blueprint of a design used for solving a specific problem

Advantages of flowchart

- Flowchart is excellent way of communicating the logic of a program.
- Easy and efficient to analyze problem using flowchart.
- During program development cycle, the flowchart plays the role of a blueprint, which makes program development process easier.
- After successful development of a program, it needs continuous timely maintenance during the course of its operation. The flowchart makes program or system maintenance easier.
- It is easy to convert the flowchart into any programming language code.

Flowchart symbols (1/2)

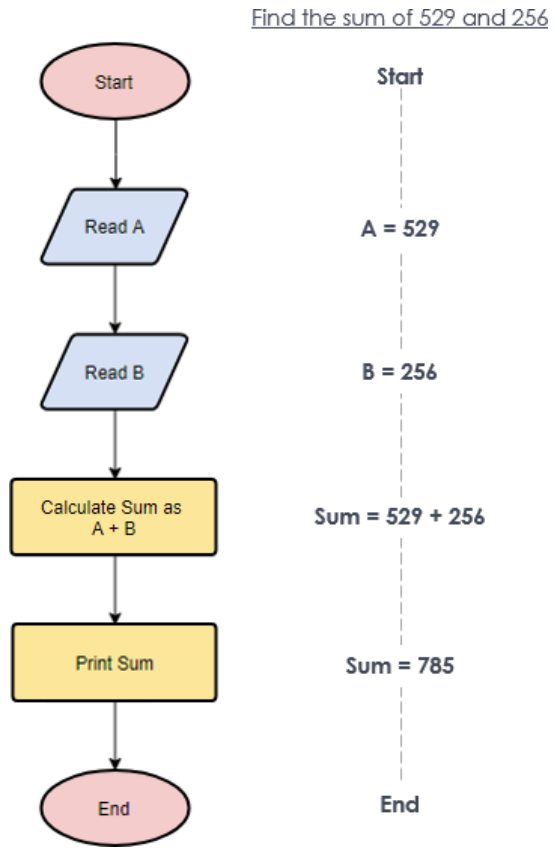
	Represents Start, End
	Represents Input, Output data
	Represents Process (actions, calculations)
	Represents Decision Making
	Represents Pre-defined Process / module

Flowchart symbols (2/2)

	Represents off page connector which are used to indicate that the flow chart continues on another page. Page numbers are usually placed inside for easy reference.
	**Connector Symbol** represents the exit to, or entry from, another part of the same flow chart. It is usually used to break a flow line that will be continued elsewhere.
	The **Document Symbol** is used to represent any type of hard copy input or output (i.e. reports).
	Represents control flow

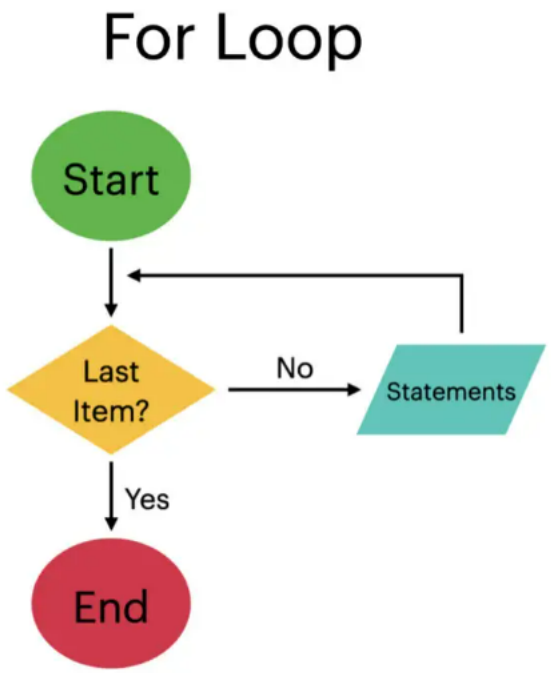
Some control structure

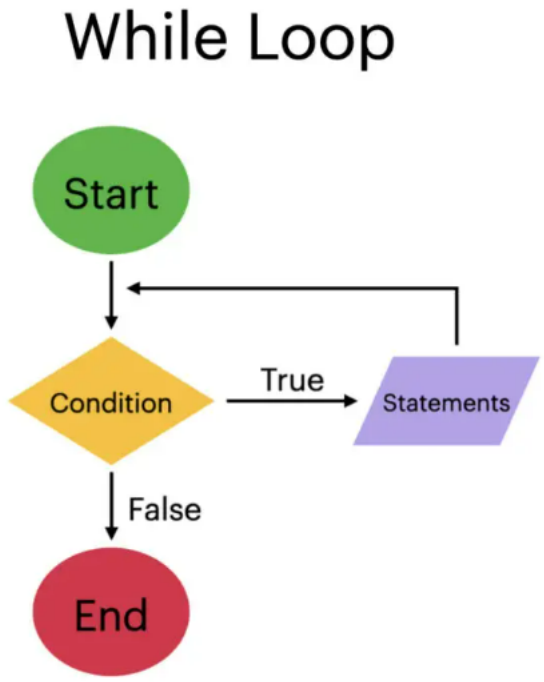
- Sequence



Some control structure

- Loop



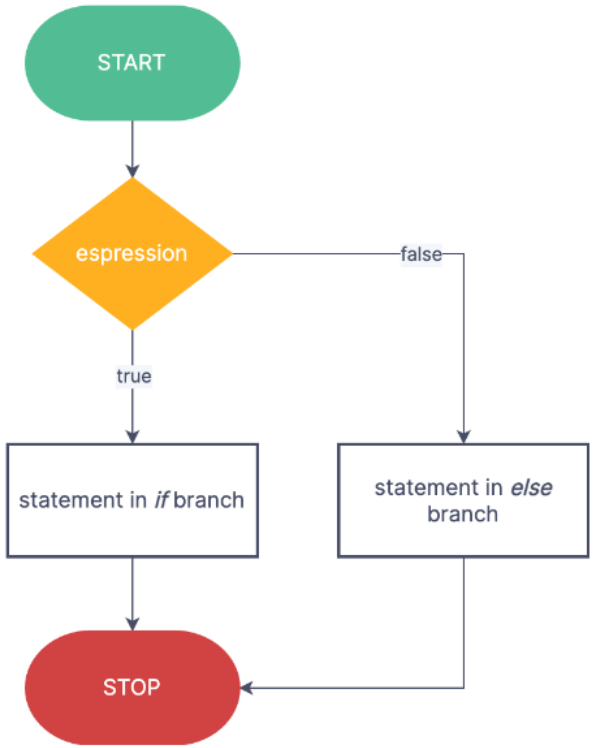


BTEC Alliance with FPT Education

Pearson

Some control structure

- Selecting



If-else Flowchart

BTEC Alliance with FPT Education

Pearson

Activity 1

1. Ask students to consider everyday tasks with a sequence of steps (for example, preparing to play a game using PlayStation, making a sandwich, fixing a puncture on a bicycle tyre, making a cake).
2. Ask students to suggest a sequence of steps for cleaning teeth. Remind them to sequence the steps in a logical order.

Activity 2

- **Give the class a list of algorithms in Math**

EX: Algorithm for addition:

- Step 1: Line up the numbers vertically by matching the place values.
- Step 2: Add together the numbers that share the same place value, starting with the ones column.
- Step 3: Write the sum below each column.
- Step 4: If the sum of a column is greater than 9, carry over the tens digit to the next column.

Pseudo code

- Pseudo code is basically short English phrases used to explain specific tasks within a program's algorithm. It should not contain any specific computer languages.
- If you can not write it in Pseudo code you won't be able to write it in C++ or Java

Pseudo code

- IF <condition> THEN …ENDIF
- IF <condition> THEN … ELSE … ENDIF
- WHILE <condition> DO … ENDWHILE
- DO … UNTIL <condition>
- DISPLAY …
- RETURN

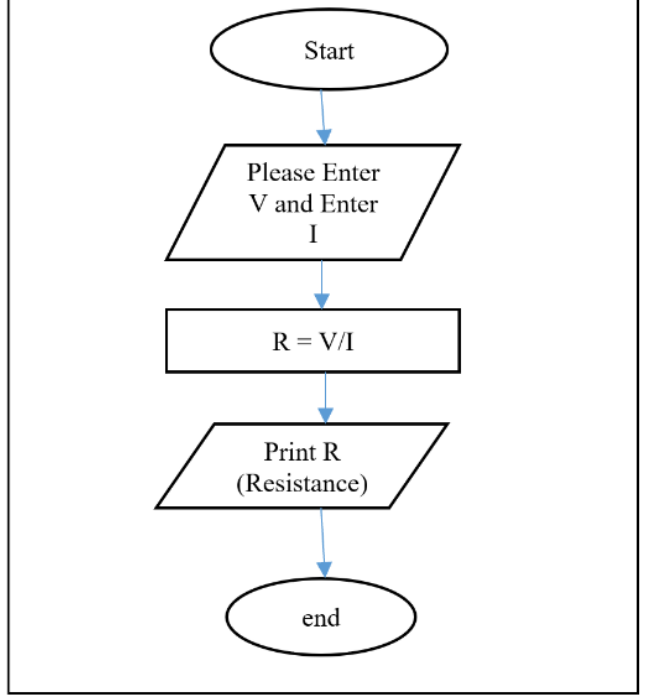
Pseudo code

Example 1: Develop a pseudocode and a flowchart to ask the user to enter the voltage (V) and Current (I), then calculate and displays the resistance R = V/I.

Pseudocode:

```
Start
        PRINT "Please enter the Voltage"
        GET V
        PRINT "Please enter the Current"
        GET I
        R = V/I
        PRINT "Resistance = "
Stop
```

Flowchart:



BTEC Alliance with FPT Education

Pearson

Algorithm Analysis

To compare algorithms, let us define a *few objective measures:*

- **Execution times?** *Not a good measure* as execution times are specific to a particular computer.
- **Number of statements executed?** *Not a good measure*, since the number of statements varies with the programming language as well as the style of the individual programmer.
- **Ideal solution?** Let us assume that we express the running time of a given algorithm as a function of the input size *n* (i.e., *f*(*n*)) and compare these different functions corresponding to running times. This kind of comparison is independent of machine time, programming style, etc.

Algorithmic Performance

There are *two aspects* of algorithmic performance:

- Time
  - Instructions take time.
  - How fast does the algorithm perform?
  - What affects its runtime?
- Space
  - Data structures take space
  - What kind of data structures can be used?
  - How does choice of data structure affect the runtime?

➢We will focus on time:

- How to estimate the time required for an algorithm
- How to reduce the time required

Summary

- Draw a mind-map to summarize the content of this lecture