Chapter 1

INTRODUCTION TO COMPUTER AND PROGRAMMING
Chapter 1

- Hardware and software
- Programming Languages
- Problem solution and software development
- Algorithms
Computer Hardware

- Input unit
- Output unit
- Memory unit
- ALU
- CPU
- Secondary storage
Input Unit and Output Unit

- **Input Unit**
  - It obtains information from various *input devices* and places this information at the disposal of the other units.
  - Examples of input devices: keyboards, mouse devices.

- **Output Unit**
  - It takes information that has been processed by the computer and places it on various *output devices*.
  - Most output from computer is displayed on screens, printed on paper, or used to control other devices.
Input Unit and Output Unit

Standard Input (keyboard) → Program → Standard Output (screen)

Input-output

input → black box → output

feedback
Memory Unit

- The memory unit stores information. Each computer contains memory of two main types: RAM and ROM.

- **RAM** (*random access memory*) is volatile. Your program and data are stored in RAM when you are using the computer.

- **ROM** (*read only memory*) contains fundamental instructions that cannot be lost or changed by the user. ROM is non-volatile.
ALU and CPU

- Arithmetic and Logic Unit (ALU)

  ALU performs all the arithmetic and logic operations.
  Ex: addition, subtraction, comparison, etc..

- CPU

  The unit supervises the overall operation of the computer.
Secondary Storage

- Secondary storage devices are used to be permanent storage area for programs and data.

- Examples: magnetic tapes, magnetic disks and optical storage CD.
  - Magnetic hard disk
  - Floppy disk
  - CD ROM
  - etc..
Some terminology

- A **computer program** is a set of instructions used to operate a computer to produce a specific result.

- Writing computer programs is called **computer programming**.

- The languages used to create computer programs are called **programming languages**.

- **Software** means a program or a set of programs.
Machine languages

- Machine languages are the lowest level of computer languages. Programs written in machine language consist of 1s and 0s.

- Programs in machine language can control directly to the computer’s hardware.

- Example:

  00101010 000000000001 000000000010
  10011001 000000000010 000000000011

  opcode  address parts
A machine language instruction consists of two parts: an instruction part and an address part.

- The instruction part (opcode) tells the computer the operation to be performed.
- The address part specifies the memory address of the data to be used in the instruction.
Assembly languages

- Assembly languages perform the same tasks as machine languages, but use **symbolic names** for opcodes and operands instead of 1s and 0s.

```
LOAD BASEPAY
ADD OVERPAY
STORE GROSSPAY
```

- An assembly language program must be **translated** into a machine language program before it can be executed on a computer.

```
i = j + k;
if (i == 3)
k = 0;
else
j = j - 1;
```

```
iLOAD j  // i = j + k
ILOAD k
IADD
ISTORE i
if (i < 3)
ILOAD i
BIPUSH 3
IF_ICMP EQ L1
0x15 0x02
0x15 0x03
0x60
0x36 0x01
0x15 0x01
0x10 0x03
0x9F 0x00 0xD
0x15 0x02
0x10 0x01
0x64
0x36 0x02
0xA7 0x00 0x07
BIPUSH 0  // k = 0 0x10 0x00
ISTORE k
0x36 0x03
```

(a) (b) (c)
Assembler

Assembly language program → Translation program (assembler) → Machine language program
High-level Programming Languages

- High level programming languages create computer programs using instructions that much easier to understand.

- Programs in a high-level languages must be translated into a low level language using a program called a **compiler**.

- A compiler translates programming code into a low-level format.
High-level Programming Languages (cont.)

- High-level languages allow programmers to write instructions that look like every English sentences and commonly-used mathematical notations.
- Each line in a high-level language program is called a statement.
- Example:
  \[ \text{Result} = (\text{First} + \text{Second})*\text{Third} \]
Application and System Software

- Two types of computer programs are: application software and system software.
- *Application software* consists of those programs written to perform particular tasks required by the users.
- *System software* is the collection of programs that must be available to any computer system for it to operate.
Examples of system software

- The most important system software is the *operating system*.
  - MS-DOS, UNIX, MS WINDOWS, MS WINDOWS NT

- Many operating systems allow user to run multiple programs. Such operating systems are called *multitasking systems*.

- Beside operating systems, *language translators* are system software.
PROGRAMMING LANGUAGES

- **Some well-known programming languages:**
  - FORTRAN 1957
  - COBOL 1960s
  - BASIC 1960s
  - PASCAL 1971 Structure programming
  - C
  - C++ Object-oriented programming
  - Java

- **What is Syntax?**

  A programming language’s syntax is the set of rules for writing correct language statements.
The C Programming Language

- In the 1970s, at Bell Laboratories, Dennis Ritchie and Brian Kernighan designed the C programming language.

- C was used exclusively on UNIX and on mini-computers. During the 1980s, C compilers were written for other platforms, including PCs.

- To provide a level of standardization for C language, in 1989, ANSI created a standard version of C, called ANSI C.

- One main benefit of C: it is much closer to assembly language other than other high-level programming languages.

- The programs written in C often run faster and more efficiently than programs written in other high-level programming language.
The C++ Programming Language

- In 1985, at Bell Laboratories, Bjarne Stroustrup created C++ based on the C language. C++ is an extension of C that adds object-oriented programming capabilities.

- C++ is now the most popular programming language for writing programs that run on Windows and Macintosh.

- The standardized version of C++ is referred to as ANSI C++.

- The ANSI standards also define *run-time libraries*, which contains useful functions, variables, constants, and other programming items that you can add to your programs.

- The ANSI C++ run-time library is called Standard Template Library or Standard C++ Library.
Structured Programming

- During 1960s, many large softwares encountered severe difficulties. Software schedules were late, costs exceeded budgets and finished products were unreliable.
- People realized that software development was a far more complex activity than they had imagined.
- Research activity in the 1960s ⇒ *Structured Programming*.
- It is a discipline approach to writing programs that are clearer than unstructured programs, easier to test and debug and easier to modify.
- Pascal (Niklaus Wirth) in 1971.
  - Pascal was designed for teaching structured programming in academic environments and rapidly became the preferred programming languages in most universities.
Object Oriented Programming

- In the 1980s, there is another revolution in the software community: **object-oriented programming**.

- Objects are **reusable** software components that model items in the real world.

- Software developers are discovering that: using a modular, object-oriented design and implementation approach can make software development much more productive.

- OOP refers to the creation of reusable software objects that can be easily incorporated into another program.
Object Oriented Programming (cont.)

- An **object** is programming code and data that can be treated as an individual unit or component.

- **Data** refers to information contained within variables, constants, or other types of storage structures. The procedures associated with an object are referred as **functions** or **methods**.

- Variables that are associated with an object are referred to as **properties** or **attributes**.

- OOP allows programmers to use programming objects that they have written themselves or that have been written by others.
PROBLEM SOLUTION AND SOFTWARE DEVELOPMENT

- Software development consists of three overlapping phases
  - Development and Design
  - Documentation
  - Maintenance

- Software engineering is concerned with creating readable, efficient, reliable, and maintainable programs and systems.
Phase I: Development and Design

The first phase consists of four steps:

1. **Analyze the problem**
   Analyze the problem requirements to understand what the program must do, what outputs are required and what inputs are needed.

2. **Develop a Solution**
   We develop an algorithm to solve the problem. *Algorithm* is a sequence of steps that describes how the data are to be processed to produce the desired outputs.

3. **Code the solution**
   This step consists of translating the algorithm into a computer program using a programming language.

4. **Test and correct the program**
Phase II: Documentation

- Documentation requires collecting critical documents during the analysis, design, coding, and testing.
- There are five documents for every program solution:
  - Program description
  - Algorithm development and changes
  - Well-commented program listing
  - Sample test runs
  - User’s manual

CuuDuongThanCong.com
https://fb.com/tailieudientucntt
Phase III: Maintenance

- This phase is concerned with
  - the ongoing correction of problems,
  - revisions to meet changing needs and
  - the addition of new features.
ALGORITHMS

- You can describe an algorithm by using flowchart symbols. By that way, you obtain a flowchart.

- **Flow chart** is an outline of the basic structure or logic of the program.

- Another way to describe an algorithm is using pseudocode.

- Since flowcharts are not convenient to revise, they have fallen out of favor by programmers. Nowadays, the use of pseudocode has gained increasing acceptance.
Flowchart symbols

- Terminal
- Input/output
- Process
- Flowlines
- Decision
- Connector
- Predefined process
Example

Start

Input Name, Hours, Rate

Calculate
Pay ← Hours × Rate

Display
Name, Pay

End

Note:
Name, Hours and Pay are *variables* in the program.
Algorithms in pseudo-code

- You also can use English-like phases to describe an algorithm. In this case, the description is called *pseudocode*.

- **Example:**

  *Input the three values into the variables Name, Hours, Rate.*
  
  *Calculate* \( \text{Pay} = \text{Hours} \times \text{Rate} \).
  
  *Display Name and Pay.*
Loops

Note:
1. Loop is a very important concept in programming.
2. \( \text{NUM} \leftarrow \text{NUM} + 1 \) means old value of \( \text{NUM} + 1 \) becomes new value of \( \text{NUM} \).

The algorithm can be described in pseudocode as follows:

\[
\text{NUM} \leftarrow 4 \\
\text{do} \\
\quad \text{SQNUM} \leftarrow \text{NUM}^2 \\
\quad \text{Print } \text{NUM}, \text{SQNUM} \\
\quad \text{NUM} \leftarrow \text{NUM} + 1 \\
\text{while } (\text{NUM} \leq 9) \\
\text{STOP}
\]
Exercise