Chapter 1

Computer Technology
Concept of Programming
&
Algorithm and Flow Charts
**What is Computer?**

*Computer* is a device that can perform computations and make logical decisions billions of times faster than human beings can.

Many of today’s personal computers can perform several billion additions per second.

Nowadays, **almost everything** around us is somehow influenced by computer technology. We are interacting with computer technology almost every stages of our everyday life.

- Playing Games on Computers or on other devices
- GSM networks and communication systems
- Traffic lights, speeding radars, and security cameras
- Online processing (paying bills, bank transactions, or registering for courses).
- Elevators or TV systems
- Most importantly, challenging engineering problems are solved by using computer technology and computer programs.

All of these much or less uses computer technology and requires computer programming.
Traits/Features/Character of a Good Computer Programmer

Most common traits must be found in a good programmer are the followings

• Ready to challenge to solve problems
• Develop and algorithm to solve a problem and implement the idea by programming
• Being patient and ready to try different methods to solve problems
• Love and concentrate on what you do; you may need to spend hours for a narrow task of a problem
• Writing and communicating ideas is important.

Considering that the most of the computer technology is originated from west, it is good to be good in English to make sure that you are able to follow most up-to-date topics.
Overview of Computers (History)

• First Electronic Computer was built by Dr. John Atanasoff and Clifford Berry to assist the graduate studies in Nuclear Physic at Iowa State University in late 1930s.

• Back in 1949, the first large-scale general purpose computer called ENIAC (Electronic Numeral Integrator and Computer) was built at University of Pennsylvania with the support of US army. Dimensions are 30 tons 30’ by 50’. Only used for ballistic tables, weather forecast, and for atomic physics calculations.

After the developments in solid state physics and material science back in 1980s it had become possible to make for general and personal usage at smaller sizes and with faster speeds and higher performances.
Overview of Computers (Nowadays Computers)

Computers used/owned by a single person are called Personal Computers (PC). Most commonly used PCs are shown below.
Overview of Computers (Hardware and Software)

A computer can be put into two major categories

• **Hardware**: Electronic part of the computer.
• **Software**: Program part of the computer.

**Hardware**

- Input devices
- Secondary storage
- Main memory
- Central processing unit
- Output devices

**Software**

- Microsoft Windows
- Office
- Novell
- AIX
- Red Hat
- Debian
- Fedora
- Ubuntu

- Linux & Unix: everything under the sun
How Computers Work?(Bit and Byte)

Computers are the machines that process information. Smallest amount of information that can be processed by computer is called “Bit”. This one bit information is “1” or “0”.

And 8 bits of information is called “1 Byte”.

Capacity of memories and storage units are expressed in in Bytes.

\[
\begin{align*}
1 \text{ B} &= 8 \text{ b} \\
1 \text{ KB} &= 1024 \text{ B} \sim 10^3 \text{ B} \\
1 \text{ MB} &= 1024 \text{ KB} \sim 10^6 \text{ B} \\
1 \text{ GB} &= 1024 \text{ MB} \sim 10^9 \text{ B} \\
1 \text{ TB} &= 1024 \text{ GB} \sim 10^{12} \text{ B} \\
1 \text{ PB} &= 1024 \text{ TB} \sim 10^{15} \text{ B} \\
1 \text{ EB} &= 1024 \text{ PB} \sim 10^{18} \text{ B}
\end{align*}
\]
Components of a Computer (CPU)

CPU (Central Processing Unit): This “administrative” section coordinates and supervises the operation of the other sections. The CPU tells the input unit when to read information into the memory unit, tells the ALU when information from the memory unit should be used in calculations and tells the output unit when to send information from the memory unit to certain output devices. Many of today’s computers have multiple CPUs and, hence, can perform many operations simultaneously—such computers are called multiprocessors. A multi-core processor implements multiprocessing on a single integrated circuit chip—for example a dual-core processor has two CPUs and a quad-core processor has four CPUs. Speed of a CPU is expressed as Hz and 1 Hz = 1 bit/sec.
Components of a Computer (Main Memory - RAM)

Main Memory (Random-Access-Memory RAM) :
This rapid-access, relatively low-capacity “warehouse” section retains information that has been entered through the input unit, making it immediately available for processing when needed. The memory unit also retains processed information until it can be placed on output devices by the output unit. Information in the memory unit is volatile—it’s typically lost when the computer’s power is turned off. The memory unit is often called either memory or primary memory.
Components of a Computer (Input Units)

Input Units:
This “receiving” section obtains information (data and computer programs) from input devices and places it at the disposal of the other units so that it can be processed. Humans typically enter information into computers through keyboards and mouse devices. Information also can be entered in many other ways, including by speaking to your computer, scanning images and barcodes, reading from secondary storage devices (like hard drives, CD drives, DVD drives and USB drives—also called “thumb drives”) and having your computer receive information from the Internet (such as when you download videos from YouTube™, e-books from Amazon and the like).
Components of a Computer (Output Units)

Output Units:

This “shipping” section takes information that the computer has processed and places it on various output devices to make it available for use outside the computer. Most information that is output from computers today is displayed on screens, printed on paper, played on audio players (such as Apple’s popular iPods), or used to control other devices. Computers also can output their information to networks, such as the Internet.
Components of a Computer (Storage Units)

Secondary Storage Units:
This is the long-term, high-capacity “warehousing” section. Programs or data not actively being used by the other units normally are placed on secondary storage devices (e.g., your hard drive) until they’re again needed, possibly hours, days, months or even years later. Therefore, information on secondary storage devices is said to be persistent—it is preserved even when the computer’s power is turned off. Secondary storage information takes much longer to access than information in primary memory, but the cost per unit of secondary storage is much less than that of primary memory. Examples of secondary storage devices include CDs, DVDs and flash drives (sometimes called memory sticks), which can hold hundreds of millions to billions of characters.
Computer Component Performances vs. Year

Beside capacities of the memories also the speeds of these parts are significantly increasing in time. As one may notice from the following graph, CPU speeds are increasing the most during the last decades.
Hard Drive Prices vs. Year

We are lucky that the prices for the computer parts are going down every year. As an example, below graph shows the hard drive cost per 1 GB in US $.

Can you imagine buying a 1TB hard drive back in 1980? Price is 100 Million $.
Overview of Computers (Software - OSs)

Software is the program part of a computer. The software serves as a bridge between the user and the computer hardware. The computer is useless without software. Main software on a computer is the operating system (OS). Most commonly used OSs are the followings.

**Command-Line Interface**
- UNIX
- MS-DOS
- VMS

**Graphical User Interface**
- Macintosh OS
- Windows
- Unix +X Window System
- Linux

Nowadays, most of the modern PCs are using Windows OS and also Apple Brand Devices are using Macintosh OS.

Most common smartphones in the marketplaces are using Android, iOS and Windows operating systems. Android is Linux based operating system whereas, iOS is Macintosh based OS.
Overview of Computers (Software - Applications)

Applications are secondary-software in a computer and for more specific purposes. For instance MS-office is designed for office documentation whereas AUTOCAD is designed for engineering drawing tasks.
Computer Networks (LAN)
As we have learned that the data/information on a computer is kept on secondary memory (hard drives). It is possible to connect several computers and make them communicate with one another. In this way, it becomes possible to share information between computers. This is called “Computer Network”.

The computer network between different devices can be done via cable or wireless connection. The small size computer networks at home, at small business, or at school is called “Local Area Network (LAN)”. 
Computer Networks (WAN)

Larger networks including the computers from Government, Security, etc. services and satellite systems are called “Wide Area Network (WAN)”.

www is the abbreviation for World Wide Web is the internet WAN. This enables us to reach data stored on other computers called servers.
Computer Program and Programming

All of these mentioned computer technology requires programming. Making a program requires programming. Basically a programmers writes programs such that the computer does what the user want.

What is a computer program?
A computer program is a list of instructions given to computer. The CPU executes these instructions then the computer does what we want.

How can we give instructions to a computer?
In order to instruct computer, we must instruct in the way that a computer can understand. So we need to speak/talk the same language that the computer understand, this is why the programming languages exist.

There are two types of programming languages:
• Low-Level Programming Languages (LPL)
• High-Level Programming Languages (HPL)

The languages has advantages and disadvantages.
Types of Computer Programming Languages

Low-Level Programming Languages

This type of instructions are in machine languages. In LPL the instructions given to CPU are in terms of bits 01001... These instructions are CPU dependent so varies from CPU to CPU. It is quite hard to learn and program. Therefore it is not the preferred by the programmers most of the times.

High-Level Programming Languages

In HPL, the instructions are similar to real language (mostly English) like read, print scan etc. In High-Level programming a compiler is needed to translate/convert such instructions to the machine language which can be understood by the computer, CPU. The drawback of HPL is they are not as fast as LPL. The advantage of HPL is that the programs are not CPU dependent so these can be used on any computer CPU.

In this lecture we will learn C programming language which is a HPL.
## Most Commonly Used HPLs

Below table represents the some of the most commonly used HPLs.

<table>
<thead>
<tr>
<th>Language</th>
<th>Application Area</th>
<th>Origin of Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORTRAN</td>
<td>Scientific programming</td>
<td>Formula translation</td>
</tr>
<tr>
<td>COBOL</td>
<td>Business data processing</td>
<td>Common Business-Oriented Language</td>
</tr>
<tr>
<td>LISP</td>
<td>Artificial Intelligence</td>
<td>List processing</td>
</tr>
<tr>
<td>C</td>
<td>Systems programming</td>
<td>Predecessor language was named B</td>
</tr>
<tr>
<td>Prolog</td>
<td>Artificial Intelligence</td>
<td>Logic programming</td>
</tr>
<tr>
<td>Ada</td>
<td>Real-time distributed systems</td>
<td>Ada Augusta Byron collaborated with nineteenth-century computer pioneer Charles Babbage</td>
</tr>
<tr>
<td>Smalltalk</td>
<td>Graphical user interfaces; object-oriented programming</td>
<td>Objects “talk” to one another via messages</td>
</tr>
<tr>
<td>C++</td>
<td>Supports objects and object-oriented programming</td>
<td>Incremental modification of C (++ is the C increment operator)</td>
</tr>
<tr>
<td>Java</td>
<td>Supports Web programming and programming Android applications</td>
<td>Originally named “Oak”</td>
</tr>
</tbody>
</table>
High Level Programming and Compile

Below diagram shows how a program written in HPL is compiled.

1. **Word Processor** (editor) Used to type in program and corrections
2. **Source File** Format: text
3. **Compiler** Attempts to translate program into machine code
   - Successful
   - Unsuccessful
     - **Error Messages**
4. **Object File** Format: binary
5. **Other Object Files** Format: binary
6. **Linker** Resolves cross-references among object files
7. **Executable File** (load module) Format: binary
8. **Loader** Copies executable file into memory; initiates execution of instructions
9. **Input data**
10. **Results**
Concept of Programming

Before we learn how to write a program, we will first learn the concept/the idea of programming. There are 3 States in Developing a program. As good programmer these stages must be followed.

Stages of Developing a Program

1. **Designing the Program:** Decide how to do it. We also call this stage as developing an algorithm.

2. **Implementation Stage of the Algorithm:** In this stage we start writing the program.

3. **Check/Test Stage:** We check/test the program if it is doing what we want by using arbitrary and different cases.
Concept of Programming (Example Algorithm)

Algorithm for a robot to get out of the classroom. Consider a robot placed in a classroom close to the room. Design an algorithm for a program which guides the robot to get out of room. You may use instructions like **start, stop, walk, if the door open, open the door** and **close the door**.

Algorithm
1. Start
2. Walk 5m
3. If the Door is open Walk out, Close the door. Stop
4. Open the Door Walk out, Close the door. Stop
Concept of Programming (Flow Charts)

In the first stage of programming, we design our program. As beginner, it is easier to express the program by using flow-charts. Flow Charts are good way of showing a program visually. After getting experience, you don’t need to use the flow charts to begin with.

Below is a flow-chart for a program that prints the largest of three numbers.
Elements in Flow Charts

Most commonly used elements in flow charts and some others are shown below.

**Commonly Used Elements**

- Start / End
- Computation / Assignment
- Input / Output Operations
- Decision Making and Braching
- Connector or Joining of Two Part in a program.
- Loops / Repetitions
- Flow Lines – Shows the direction to proceed.

**Rarely Used Elements**

- Magnetic Disk (Secondary Memory)
- Output to Printer
- Magnetic Tape (Floppy)
Example 1: Printing (Algorithm and Flow-Chart)

Write the algorithm and draw flow-chart for a program. The program prints the “Hello World!” to the screen.

**Algorithm**

1. Start
2. Print “Hello World!”
3. End

This is the simplest program one can think. When we start writing program, as we will see later there will be a command/statement corresponding each element, so that we will be easily write the program based on the flow-charts.

For example, printing command in C programming language is **printf**
Example 2: ..... (Algorithm and Flow-Chart)

Write the algorithm and draw flow-chart for a program. The program determines if a given real number is “zero”, “positive”, or “negative” and write this information to the screen. **

Algorithm
1. Start
2. Prompt user to enter a number
3. Read a number and store it in A
4. If A= 0 print “ZERO” then DONE!
5. If A > 0 print “POSITIVE” then DONE!
6. print “NEGATIVE” then DONE!
7. Finish

Now try testing you program by using 10 , 0 and -10
Example 3: ..... (Algorithm and Flow-Chart)

Write the algorithm and draw flow-chart for a program. The program takes two numbers from keyboard then writes the sum and product of these numbers to the screen. **

**Algorithm**

1. Start
2. Prompt user to enter two numbers
3. Read two numbers and store them in A, B
4. Define C = A+B
5. Define D = A*B
6. Write C
7. Write D
8. Finish

Now try testing you program by using 3 and 5
Example 4: ..... (Algorithm and Flow-Chart)

Write the algorithm and draw flow-chart for a program. The program takes two numbers from keyboard then writes the larger one to the scree. If the numbers are the same then the program prints, “Numbers Same” to the screen. **

**Algorithm**
1. Start
2. Read two numbers and store them in A, B
3. If A=B then print “Numbers Same” DONE!
4. If A>B then print “Larger”, A . DONE!
5. Print “Larger”, B
6. Finish

Now try testing your program by using
7 and 11
15 and 6
5 and 5
Example 5: ..... (Algorithm and Flow-Chart)

Write the algorithm and draw flow-chart for a program. The program reads the midterm and final exam results of a student then calculates the final grades as 40% of the midterm and 60% of final exam. Then writes to the screen “PASSED” if the grade is equals or greater than 50 or “FAILED” if grade is less than 50. **

Algorithm
1. Start
2. Read two numbers and store them in M, F
3. Assign/Calculate G = M*.4+F*.6
4. If G ≥ 50 then print “PASSED” , DONE!
5. Print “FAILED”
6. STOP/FINISH/END

Now try testing you program by using 40 and 60  
20 and 50
**Example 6: Quadratic Eq. (Algorithm and Flow-Chart)**

Write the algorithm and draw flow-chart for a program. The program reads the coefficients of a quadratic equation in the form $ax^2 + bx + c = 0$ from the keyboard then determines the solutions and print the solution/s to the screen. If there is no real solution, program should print “No Real Solution” to the screen. **

**Algorithm**

1. Start
2. Prompt user to enter the coefficients.
3. Read the coefficients $a,b,c$
4. Assign $d=b^2 - 4ac$
5. If $d < 0$ print “No Real Solution” END!
6. if $d = 0$ assign $x = -b / 2a$
   print “One Solution”, $x$
7. Assign $x_1 = \frac{-b-\sqrt{d}}{2a}$ and $x_2 = \frac{-b+\sqrt{d}}{2a}$
8. print “Two Solution”, $x_1,x_2$
9. STOP/FINISH/END

Don’t forget to test your program with different values.
Example 7: .... (Algorithm and Flow-Chart)
Write the algorithm and draw flow-chart for a program. The program inputs a number from keyboard then prints to screen “EVEN” or “ODD” if the number is even or odd respectively. Not: You may use % modulus / remainder operator. **

Algorithm
1. Start
2. Prompt user to enter a number
3. Read the number N
4. Assign mod = N % 2
5. If mod = 0 then print “EVEN” END!
6. Print “ODD”
7. STOP/FINISH/END

Test your program with inputs 6 and 11.
Example 8: ..... (Algorithm and Flow-Chart)

Write the algorithm and draw flow-chart for a program. The program inputs 3 numbers from the keyboard then prints the largest one to the screen. **

Algorithm

1. Start
2. Prompt user to enter three numbers
3. Read three numbers and store them in A,B,C
4. Compare the first two numbers, A and B
5. Compare the largest of A and B with C then print the largest to the screen
6. STOP/FINISH/END

Now try testing you program by using
5, 7, and 2
11, 3, and 21
Example 9: ..... (Algorithm and Flow-Chart)

Write the algorithm and draw flow-chart for a program. The program inputs the age the user then based on the age of the person, the program prints the age status of the person to the screen. **

Algorithm

1. Start
2. Prompt user to enter his/her age.
3. Read the entered number and store it in A variable.
4. If A ≤ 4 print “BABY” then END!
5. If A ≤ 10 print “CHILD” then END!
6. If A ≤ 18 print “TEEN” then END!
7. If A ≤ 50 print “ADULT” then END!
8. print “OLD”
9. STOP/FINISH/END

Don’t forget to test your program.
Try with 10, 25, and 59
Example 10: ..... (Algorithm and Flow-Chart)

Write the algorithm and draw flow-chart for a program. The program prints “Hello World!” to screen 10 times. **

**Algorithm**

1. Start
2. Assign C=0 (C is counter to count number of prints)
3. Print “Hello World!”
4. Assign C = C +1 (increasing the counter)
5. If C < 10 go to step 3.
6. STOP/FINISH/END
Example 11: (Algorithm and Flow-Chart)

Write the algorithm and draw flow-chart for a program. The program prints out the integer numbers from 0 to a desired positive integer number. **

**Algorithm**

1. Start
2. Prompt user to enter a number.
3. Read the entered number and store it in N variable.
4. If N < 0 go to step 2.
5. Define/Assign Counter = 0
6. Print the Counter
7. Increase Counter by +1
8. If Counter ≤ N go to step 6.
9. STOP/FINISH/END

Test your program with -5, 0, and 7

One may also do this by using loop / repetition algorithm with step of +1.
Example 11: Revision with Repetition / Loop (Flow-Chart)

Write the algorithm and draw flow-chart for a program. The program prints out the integer numbers from 0 to a desired positive integer number. *

We may also do this by using loop / repetition algorithm with step of +1.

Loop / Repetition where Counter goes from 0 to N step of +1 Repetition is N+1 times.
Example 12: .... (Algorithm and Flow-Chart)

Explain what does the following program with flow-chart does? By using a loop/repetition procedure redraw another flow-chart that does the same. **

*It prints the sum of the integers from 1 to 10.
* We can do this by using repetition step of +1
Example 13: .... (Algorithm and Flow-Chart)

Write the algorithm and draw flow-chart for a program. The program calculates the factorial (n! = 1⋅2⋅3 ... n − 1 ⋅ n) of a desired positive number and prints it to the screen **.

Algorithm
1. Start
2. Prompt user to enter a positive integer number.
3. Read the number N
4. Assign fact=1
5. Set a repetition/loop with counter I starting from 1 to N, inside the repetition, multiply fact with I.
6. Print the fact
7. STOP/FINISH/END

Test your program with input 5.
Example 14: .... (Flow-Chart)

Draw flow-chart for a program. The program determines the sum of the even numbers from 0 to a desired number. **