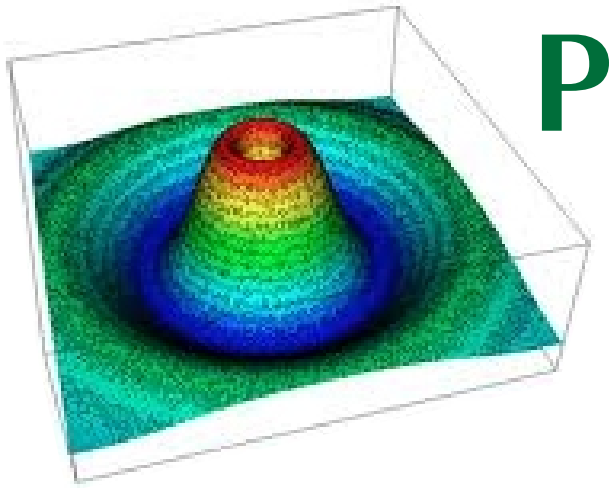
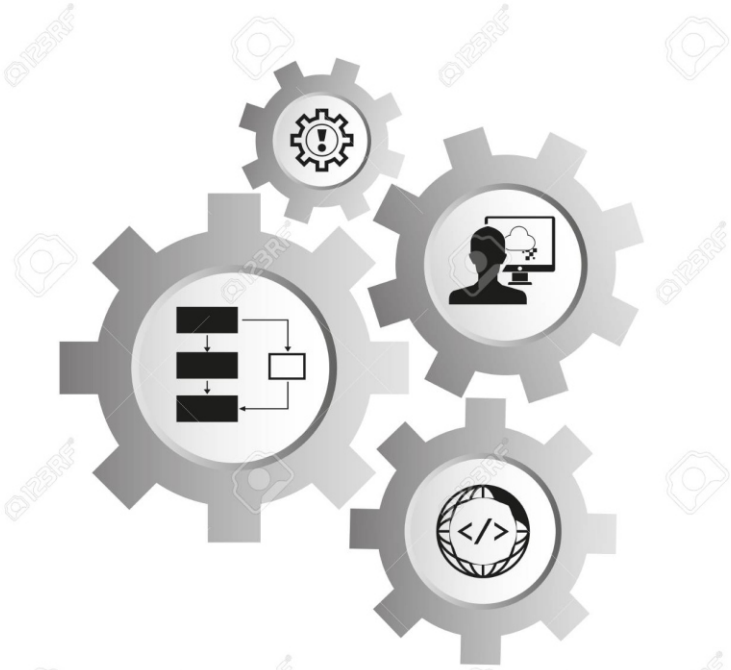
Part 1



Programming Concepts

Chapter 1



Programming Concepts

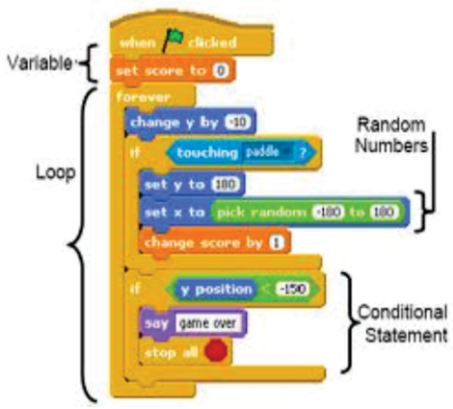
In this chapter, we will look at the differences between hardware and software and define the principal parts of a computer. Since the computer is an electronic machine, all information it uses must be reduced to electronic states. These states are represented by binary digits (or bits) Changing sounds or images to pieces of data represented by bits is the process of digitization of information. Once information has been digitized, it can be fed into a computer and manipulated. Machine language operates with bits, while other languages need to be translated into machine language so the computer can execute their commands. You'll learn about the algorithm, which is necessary in the design of solutions to problems and which programmers can code in any number of languages.

After completing this chapter you will be able to:

- tell the importance of hardware components and software;
- explain the concept of digitization;
- define binary digits, bits, bytes;
- know about computers;
- discuss computer as an electronic machine and
- differentiate high-level and low-level language in the field of computer programming.

A. Introduction

Computer technology is all around us, not only on our personal computers, but also on ATM machines, fuel injection in cars, digital cameras, and telephone communications. None of this technology would be possible without computer programming. Programming provides instructions that tell the machine how to operate. In your personal computer, everything it does - from playing video games and music to typing simple documents in Microsoft Word - requires a set of instructions.



This book will provide you with a basic to advanced concepts and knowledge of computer programming. It will tell you how to control the computer so that it can do something over and over again, make a decision, and store information in the right kind of holder, contingent on what that information looks like. These are just a few of the things that you will learn.

It is important to cover certain fundamentals before learning to program. The more you understand the machine and how it works, the easier it will be for you to grasp the concepts of programming.

B. Hardware and Software

A computer is usually hooked up to a printer, an external drive, and various other peripheral devices, such as scanners, modems, and so on. All the physical components of a computer make up its hardware. Think of the word "hardware" as describing those parts of a computer that are hard or can be touched. You can touch a printer, but you cannot touch programs that are running on a computer. Programs represent software. Any programs, whether they are commercial programs, games, word processing applications, or programs that make the computer itself operate, are all examples of software.

Programs are sets of steps that tell a computer what to do. There are directions for everything that happens in the computer. For example, saving a file on the hard drive occurs because inside the computer a program is "telling" the computer to save a file rather than delete it. Saving a file, printing a file, or deleting it is just a few of the programs called system programs. They are also referred to as the operating system. These programs allow the computer to handle its basic operations: opening and closing files and saving or deleting files. Likewise, getting an application, such as Microsoft Word, to open up and start running is the task of the operating system

Application programs are programs sold in the market, for example, WinZip, iTunes, Microsoft Office, Open Office, WordPerfect, Adobe Acrobat, Skype, and so on. The word "application" refers to a set of programs "applied" to some real-life task to make it easier to do. Some of the earliest application programs such as Microsoft Word, Word Perfect, and Claris Works were created to facilitate typing long documents.

Tip System software is the set of programs that enable the computer to store data, retrieve data, save files, delete files, and, generally, allow the computer to operate application programs.

Let's look at a list of the main parts of a computer:

- Keyboard
- Mouse
- Printer
- Hard drive
- External drive
- RAM (random access memory)
- CPU (central processing unit)

B.1. The Keyboard, Mouse, and Printer

The keyboard is used to communicate with the computer; so is the mouse. The keyboard is used for typed commands, while the mouse is used for "clicking" and interacting with the graphical user interface (GUI). The printer delivers on paper what's on the screen.

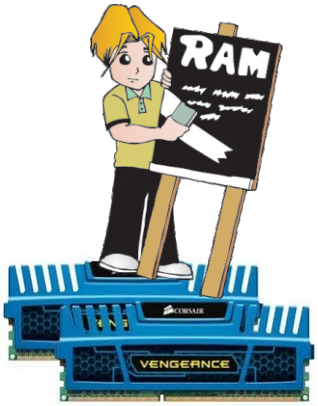
B.2. Hard Drive vs. External Drive

The hard drive is the internal storage of a computer. Application programs are saved here, as system files. Think of the hard drive as the permanent storage space a computer has. A house with a basement and an attic has much more storage area than an apartment with only closets for storage. Computers with a large hard drive, for example, 1 Terabyte (TB), are in demand because of their capacity to save very large applications. Dota(game), for example, needs at least 300 megabytes (MB) of storage space. Devices known as solid state drives can provide more portable hard memory using flash technology.

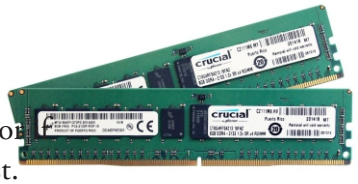
An external drive is used to expand storage capacity outside of the computer itself. As early 1990s, many personal computers had little memory in the hard drive, necessitating storage outside of the computer. Memory needed for application programs at that time was not what it is today. Although internal hard drive capacities are much greater today than in the past, there are still many uses for external drives, such as flash drives for storing pictures, video clips, and so on.

B.3 RAM: Random Access Memory

The RAM is the workspace for the computer. Think of the RAM as a large table onto which you place many different things. If you have only a small workspace, you can't open too many things at once because they won't fit on the table.



For example, if your operating system uses 128 MB(million bytes), while Microsoft Office uses 256 MB and Adobe Acrobat uses 128 MB, you could have both applications open at the same time as the operating system (a total of 512 MB) if your computer has 1 GB(gigabyte) of RAM. Now if you run a game, such as Dota 2, you'll need another 1GB just for that application. Compare that with WarCraft 3, which requires 512 MB. Again, running the operating system alone with that game brings your total to 1.28GB; and that's why you want a computer with a good size RAM, like 3 GB, if you usually play such games. Opening several applications at once and not running out of memory is desirable.

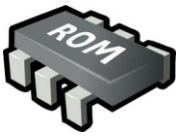


RAM is a temporary memory and is lost once the computer is turned off. The term volatile is used to describe this memory, since anything not saved will be lost in a sudden, abrupt fashion. For example, when a plug is suddenly pulled on a machine or you have to reset your computer for some reason, everything in the RAM will be lost.

Fortunately, there is always a "backup plan" for storage on a computer and this is called a swap file. The swap file is a memory taken from the hard disk space as well as from the RAM in case of an emergency. It is used only in the event that the RAM becomes exhausted and some emergency storage is needed.
The image(top right) are the latest memory sticks used in most of the personal computers of today.

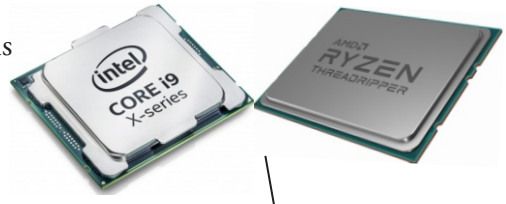
B.4. ROM: Read-Only Memory

ROM, or read-only memory, is a memory that's not lost when the machine is turned off. Certain instructions for the computer are permanently etched onto a chip at the time the computer is manufactured. Thus ROM is a permanent memory and is never lost.



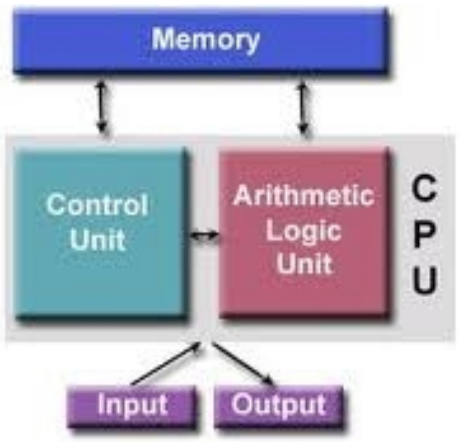
B.5. CPU: Central Processing Unit

The main part of computer is the central processing unit, or CPU. This is where the computer stores, processes, and retrieves data. The CPU manages all the functions of the computer, including processing data—manipulating data by sending it from one place to another or by performing some math on the data. The CPU contains the arithmetic/logic unit, or ALU, and the control unit of a computer. The CPU is found within the system unit and should not be confused with it. The system unit houses the internal pieces of the computer like the motherboard, the internal memory, and so on.



On your right are the latest processors(Intel & AMD) available on the market as of the writing of this book.

B.6. The ALU: Arithmetic Logic Unit



The ALU affects programmers the most. When you write a program for a computer, its ALU will be called into use to perform some math (the arithmetic part) or to evaluate a decision (the logic part) by the programmer. That's why we need the arithmetic/logic unit. The logic portion is the part of the unit that can handle decisions. Most interesting programs need the ability to make a decision.

The Control Unit

The control unit of the CPU is used to regulate program flow. This unit executes statements in sequence and will only repeat steps or skip steps if programmed to do so.

If you write a program putting a number like 5 into a holder called x and decide to increase the value of x from 5 to 7, the ALU will be used to do this task. If you want to print the message "I have had enough!" 250 times on a computer screen, then the control unit will have to do work because the programmer is asking the computer to do something over and over. Thus the programmer controls how long or how many times the computer does something; in this case, printing a message 250 times.

The computer can store, process, and retrieve data, and the CPU will handle that work.

C. Digitization: Using Numbers to Represent Information

To understand how the computer does what it does, you can examine a situation in the real world to see how numbers can be used to give information about that situation. Imagine four towns, roughly equidistant, as shown in Figure 1.1. A bad snowstorm has closed some of the roads between these towns while others are still open.

Fig. 1.1

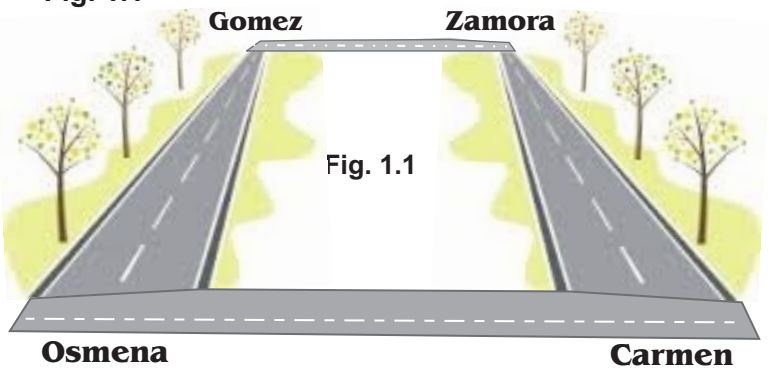


Fig. 1.2

Between Gomez and Zamora the roads are closed, as are the roads between Osmena and Gomez. The roads between Zamora and Carmen are open, and the roads between Carmen and Osmena are open. If the number one (1) represents the roads being open, and the number zero (0) represents the roads being closed, then the picture can be redrawn to look like Figure 1.2.

Now you can describe the road situation using numbers:

From	To	Road Situation	Number Description
Gomez	Zamora	closed	0
Zamora	Osmena	open	1
Osmena	Carmen	open	1
Carmen	Gomez	closed	0

The use of zeros and ones to describe a situation is a way of digitizing it. What was described with English words, like open and closed, has now been described by the numbers zero and one. These two digits are called binary digits—the word "binary" implying there are only two of them, specifically zero and one. The term bits is formed from the boldfaced letters of the two words "binary digits."

Now let's expand on the situation of the roads. What if someone from a nearby town inquires, "Can I get from Gomez to Zamora and from Zamora to Carmen?" (The towns is surrounded by mountain range, which prevents driving directly from one town to another.) The answer would be, "No, but you could go from Zamora to Carmen."

Gomez → Zamora → Carmen
Close Open

If we digitized the answer, it would look like this:

Gomez → Zamora → Carmen
0 1

If we dropped all English words, the answer would be the two bits:

0 1

For the next question, someone asks, "Can I go from Osmena to Zamora via Carmen? The answer would Carmen is open, as is the road from Carmen to Zamora."

Osmena → Carmen → Zamora
open open

Now we digitize the answer:

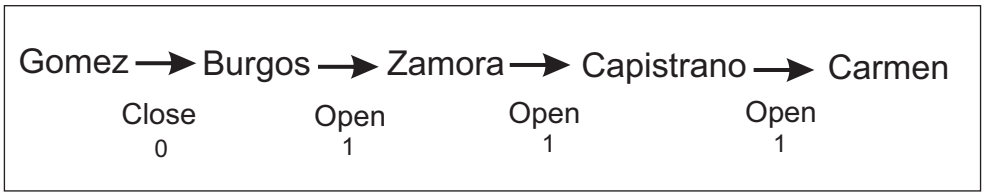
1 1

Fig. 1.3

The first 1 represents that the road is open from Osmena to Carmen, and the second 1 represents the road being open from Carmen to Zamora. Now look at an expanded map of the area, shown in Figure 1.3, this time adding the names of smaller towns between the larger towns.

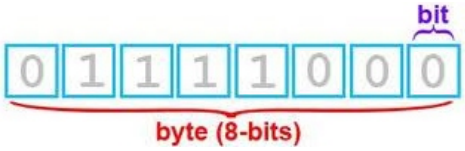
From	To	Road Situation	Number Description
Gomez	Burgos	closed	0
Burgos	Zamora	open	1
Zamora	Capistrano	open	1
Capistrano	Carmen	open	1
Carmen	Tiano	open	1
Tiano	Osmena	closed	0
Osmena	Velez	closed	0
Velez	Gomez	closed	0

A description of the road situation from Gomez to Carmen via Zamora would look like the following:



or just the four bits: 0 1 1 1

If you wanted to describe the road situations starting at Gomez and going through Burgos, Zamora, Capistrano, Carmen, Tiano, Osmena, Velez and back to Gomez, you would be looking at these bits:

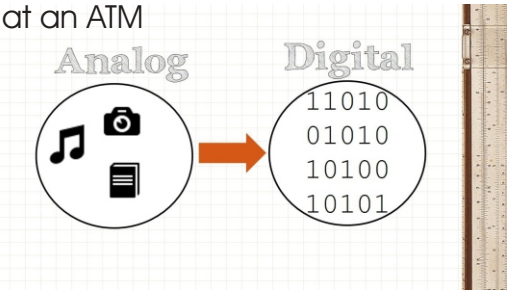


Any sequence of 8 bits is called a byte. So the sequence we just used is a byte of information. Binary digits, or bits, can be used to describe many situations in the real world aside from the road conditions mentioned.

When these numbers or digits are used to represent a particular situation or bit of information, we have digitized information

What other examples are there of digitization? We digitize photos when we scan them—all the colors used to create images that we recognize are represented digitally. The music you hear on a DVD has been digitized because musical notes can be associated with numbers as well. A password for a person using a bank card at an ATM can be digitized.

Note: Digitizing information means using numbers to represent something other than a number. We digitize sound, color, and even images. Once digitized, they can be handled by a machine that can recognize these two integers: 1 and 0.

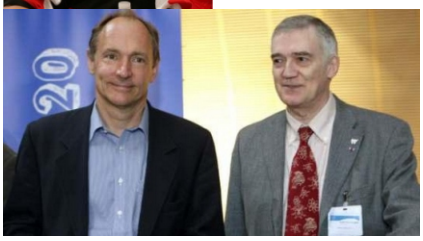


D. The Computer: an electronic machine

Richard Brodie - was the programmer of Microsoft Word (1983)

In order to be a good programmer, it is important to start with some of the fundamentals on how the computer operates. You must understand what kind of machine you're using and why it can do what it does.

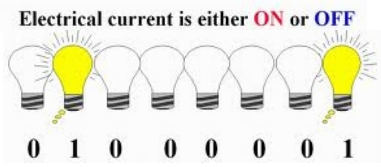
The computer is an electronic machine. Thus it needs electricity to operate. The electricity comes from batteries or plug in a wall and travels through elaborate circuitry etched onto highly sensitive material (like silicon) on a chip.



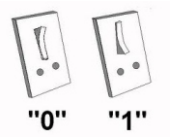
Tim Berners-Lee(left) invented/programmed what we know as the *World Wide Web* with the help of Robert Cailliau(right).

D.1. Electricity: On or Off

By manipulating electricity through the circuitry on a chip, we can go back and forth between two states: electricity flowing(ON) and electricity stopped (OFF).



If we assign the zero bit (0) to the stopped state and the one bit (1) to the flowing state, these bits can be connected to the internal workings of the machine. The machine's capacity in manipulating electronic states is used to create a language of zeros and ones called machine language. It is called machine language because it is the elementary language of the machine. Without getting bogged down in the details, the computer expresses all information with these bits and bytes of machine language.



E. Computer Languages

Languages are used in everyday life for communication. All of us speak a native language and, depending on how much we read or write, we can develop a deep understanding of that language. The important thing about language is that you use it effectively to communicate your intentions, needs, wishes, or feelings.

Computer languages are similar to spoken languages, you must use them very precisely so that you will not be misunderstood by the computer. Each language has its own grammar or syntax, which must be followed for the computer to understand that language.

Consider these examples from spoken languages:

English:	Hello, how are you?	Japanese:	Konnichi wa. O genki desu ka?
French:	Bonjour! C¸ a va bien?	Spanish:	Hola, ¿cómo estás?
German:	Guten tag. Wie geht's?		

All of these examples mean the same thing: each sentence has a greeting followed by a question asking how you are. But each example is a completely different group of words. Unless you know these languages, you would'nt know that each means the same thing.

Computer languages are similar, there are basic tasks that any computer language must do for a programmer. The programmer must learn to "speak" the language. One advantage of a computer language over a spoken language is that it does not take that long to become fluent in a computer language! Many programmers learn several languages during their careers.

Now read these examples from some computer languages:

```
BASIC    if (x > 5) print "greater."
Pascal   if x > 5 then writeln (greater.);
C++      if (x > 5) cout << "greater.";
Java     if (x > 5) System.out.println ("greater.");
```

All of these statements accomplish the same task: if the contents in the variable called x is greater than the number 5, then we will print a message on the computer screen—the word "greater."

E.1. Levels of Language: High and Low

All programming languages need to be translated into machine language, the native language of the computer. Machine language is comprised of binary code and therefore is extremely tedious to read and understand. Since the computer can only execute commands that have been written in its native machine language, other languages must be translated into machine language before being understood by the computer. There are two levels of language among programming languages: high-level languages and low-level languages.

High-Level Languages

High-level languages are languages that are "high" above the language of the computer. The machine language is the computer language it understand. High-level languages are removed from the reality of how computers process data that is, in terms of bits. They are as removed from machine language as a king or queen is removed from the everyday worries of life. Like making money, paying bills, and so on. Machine language is made of all binary digits, but high-level languages use English words to give directions to the computer. In order to understand the phrase high-level, consider first the following analogies

Analogy 1

Let's say you want to have a party for over 100 people. A "high-level" party giver would simply pick a date for the party, invite friends via e-mail or telephone, and then the party would take place. A "low-level" party giver would have to handle all the details, that is renting a place, hiring a caterer, a band, and so forth. The party would not just "happen." You would have to deal with all the details, which the well-heeled "high-level" party giver can ignore.

Analogy 2

Suppose you need to get your car fixed. A "high-level" approach to repair would be to bring the car to the repair shop and pick it up when the work is done. A "low-level" approach would be to lift the hood yourself and examine the parts, looking for the problem. Once the problem part was found, you would have to repair it yourself and then close the hood.

These analogies apply to programmers as well. The high-level language programmer doesn't need to know anything about how the computer itself goes about its work. He states a problem at a higher level without being involved in the essential details of how the computer performs its tasks. For this reason, high-level languages use commands that relate directly to the problem being programmed as opposed to the internal workings of the machine.

Programs written in high-level languages run more slowly on the computer because these languages need to be translated into machine language. Languages are defined in terms of their proximity to machine language: the higher the level, the more translation required before the program can be executed by the machine. Programs written in a high-level language require fewer lines of code than those written in a low-level language. It's much easier to say "let's give a party" or "fix the car" than say "call the caterer, order the food, rent the hall, and so on." Scratch, Visual BASIC, C/C#, Perl, and Java are some examples of high-level languages.

Note: Computers "understand" machine language, which is generated from the electronic states determined by the current and circuitry of the machine.

E.2 Low-Level Languages

Low-level languages are just above machine language level. As such, they do not undergo as much translation as the high-level languages. They are, however, more difficult to understand because they rely on a greater understanding of the internal workings of the machine. Assembly languages are low-level languages. To a BASIC programmer, C might be considered a low-level language, since it allows the programmer to have more control at a lower level than the BASIC programmer.

E.3 Language Helpers: Translators

Translators break down high-level and low-level language code into machine language understood by the particular processor in the CPU. There are two kinds of translators: interpreters and compilers.

E.4 Interpreters and Compilers

Translators can work in two different ways: as interpreters or compilers. Interpreters will translate one line of code at a time and generate error messages immediately. Compilers translate an entire file of code all at once, rather than line by line. The compiler will not generate error messages until all code has been translated. The original file or program that the programmer writes is called source code. Object code is the result of translation and is the machine language version of the original file. C++ is an example of a compiled language, while BASIC is an interpreted one.

Note: Translators change high-level language or low-level language into machine language that is readily understood by the computer's processor.