**Chapter I**

**Introduction to Computer Science**

**Chapter I Topics**

1.1 How Do Computers Work?

1.2 Messages with Morse Code

1.3 Electronic Memory

1.4 Memory and Secondary Storage

1.5 What Is Programming?

1.6 Programming Languages

1.7 Networking

1.8 Hardware and Software

1.9 Summary

**1.1 How Do Computers Work?**

Human beings do not spend money on expensive items unless such items somehow improve human capabilities. Cars are great. They move faster than humans, they do not get tired, and they keep you comfortable in bad weather. They are expensive, but the expense is worth it. Computers process information and do this processing better in many areas compared to human beings. The three areas in which computers are superior to human beings are shown in figure 1.1.

**Figure 1.1**

|  |
| --- |
| **Three Areas Where Computers Beat People** |
| • Computers are faster  • Computers are more accurate  • Computers do not forget |

You may be quick to accept that *computers are faster*, but you are not so sure about the other two. Too often you have heard the term *computer error*and you also remember hearing about data that was lost in the computer.

Well, let us start our computer lesson right now by clearing up some basic myths. *Computers do not make errors.*  Sure, it is possible for a computer to give erroneous information. However, the computer is nothing but a machine that faithfully, and always accurately, follows instructions. If the instructions given by a human being to a computer are faulty, then the computer will produce errors. At the same time, many so-called *computer errors*are caused by sloppy data entry. A person who receives an outrageous electric bill is told that the computer created an erroneous bill. True, the computer printed the bill, but not until a data-entry clerk had slipped an extra zero in the amount of electricity used for the previous month.

Perhaps you are still not convinced. After all, what about the situation when a computer breaks down? Won’t that cause problems? Broken computers will certainly cause problems. However, your computer will not work at all. Your computer applications will not work and you are stuck, but the computer does not suddenly start adding **2 + 2 = 5**.

You may also have heard that people lose their computer information because of problems with disk drives. Once again this happens, but computer users who keep their computers and diskettes in a proper environment, along with a sensible backup system, do not have such problems.

Well, you give up. No point arguing with a stupid book that cannot hear you. Fine, the computer is *faster*, the computer is *more accurate*, and sure the computer *does not forget*. But how is this managed electronically? You know that electricity is incredibly fast, and you have every confidence that the flip of a switch turns on a light or a vacuum cleaner. Computers are electronic and just how does electricity *store information*? How does a computer perform *computations*? How does a computer translate keyboard strokes into desirable output? These are all good questions and an attempt will be made here to explain this in a manner that does not become too technical.

**1.2 Messages with Morse Code**

Unless you are a Boy Scout or Navy sailor, you probably have little experience with *Morse code*. Today’s communication is so much better than Morse code, but there was a time when Morse code was an incredible invention and allowed very rapid electronic communication.

Imagine the following situation. Somehow, you have managed to connect an electric wire between the home of your friend and yourself. You both have a buzzer and a push button. Each one of you is capable of “buzzing” the other person, and the buzzer makes a noise as long as the button is pressed. You have no money for a microphone, you have no amplifier and you have no speakers. Furthermore, your mean parents have grounded you to your room without use your cell phones. But you do have your wires, your buzzers and your buttons.

Can you communicate? You certainly can communicate if you know Morse code or develop a similar system. Morse code is based on a series of **short** and **long** signals. These signals can be sounds, lights, or other symbols, but you need some system to translate signals into human communication. Morse code creates an entire set of short and long signal combinations for every letter in the alphabet and every number. Usually, a long signal is three times as long as a short signal. In the diagram, below, a long signal is shown with a bar. A short signal is indicated by a square. Consider the first five letters in the Morse code system, shown in the figure 1.2 table.

**Figure 1.2**

|  |  |  |
| --- | --- | --- |
| **First Five Letters In Morse Code** | | |
| **A** | **▀ ▀▀▀** | short - long |
| **B** | **▀▀▀ ▀ ▀ ▀** | long - short - short - short |
| **C** | **▀▀▀ ▀ ▀▀▀ ▀** | long - short - long - short |
| **D** | **▀▀▀ ▀ ▀** | long - short - short |
| **E** | **▀** | short |

You, and your buddy, can now send messages back and forth by pressing the buzzer with long and short sounds. Letters and numbers can be created this way. For instance, the word **BAD** would be signaled as follows:

▀▀▀ ▀ ▀ ▀ ▀ ▀▀▀ ▀▀▀ ▀ ▀

The secret of Morse code is the fact that electricity can be *turned on*, and it can be *turned off*. This means that a flashlight can send long and short beams of light and a buzzer can send long and short buzzing sounds. With an established code, such as Morse code, we can now send combinations of long and short impulses electronically. Very, very brief pauses occur between the **shorts** and **longs** of a letter. Longer pauses indicate the separation between letters. This basically means that electronically we can send human messages by turning electricity on and off in a series of organized pulses. Does this mean that Samuel Morse invented the computer? No, he did not get credit for starting the computer revolution, but it does serve as a simple example to illustrate how electricity can process letters by translating **on** and **off** situations into letters and numbers.

**1.3 Electronic Memory**

Fine, Morse code explains how letters can be translated into electronic impulses. This explains electronic communication, but Morse code does not store any letters. Morse code signals are sent and they are gone, followed by the next signal. If you doze off, you miss the signal and it is too bad. Luckily, somebody became clever and a special device was invented that printed **dots** (short signals) and **dashes** (long signals) on a paper tape as the message was received. Now that explains a paper memory, and perhaps you even remember something about punched computer cards, but we still have not gotten to an electronic memory.

Suppose you line up a series of light bulbs. How about picking eight bulbs? Each light bulb is capable of being turned **on** and **off**. With these **eight** light bulbs we can create **256** different combinations. Two tables are shown in figure 1.3 below. The first diagram shows **on** and **off**. The second diagram uses **1s**  and **0s**. In Computer Science, **1** means **on** and **0** means **off**.

**Figure 1.3**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **off** | **on** | **off** | **off** | **off** | **off** | **off** | **on** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **0** | **0** | **0** | **0** | **0** | **1** |

In this particular example, the **second** and **eighth** bulbs are **on**, and all the other lights are **off**. This represents only **one** of **256** different combinations. It certainly is not Morse code, but using the Morse code example, we can imagine that each one of the 256 combinations is assigned to a letter, a number, or some other type of character.

The number system you use is **base-10**. Counting and computation in base-10 is not simpler than other bases because it is base-10. It is simpler because you use base-10. Sounds confusing, does it not? In elementary school, you practiced multiplication tables in base-10. How many multiplication tables did you practice in base-5 or base-8? Not too many, right? Rumor has it that people developed a base-10 system, because of our ten fingers. Now in base-10, digits range from **0** to **9**. After the largest digit **9**, we must use two digits, like **10**, **11**, **12**, **13**, **14** etc. to count higher.

Mathematically speaking, counting and computation are possible in different bases. A number system that is very skimpy with digits is **base-2**. Only the digits **0** and **1** are used. Many digits are needed for even small numbers. The first 32 numbers in **base-2**, with the equivalent **base-10** values are shown in figure 1.4.

**Figure 1.4**

|  |  |  |  |
| --- | --- | --- | --- |
| **Base 10** | **Base-2** | **Base 10** | **Base-2** |
| 0 | 0 | 16 | 10000 |
| 1 | 1 | 17 | 10001 |
| 2 | 10 | 18 | 10010 |
| 3 | 11 | 19 | 10011 |
| 4 | 100 | 20 | 10100 |
| 5 | 101 | 21 | 10101 |
| 6 | 110 | 22 | 10110 |
| 7 | 111 | 23 | 10111 |
| 8 | 1000 | 24 | 11000 |
| 9 | 1001 | 25 | 11001 |
| 10 | 1010 | 26 | 11010 |
| 11 | 1011 | 27 | 11011 |
| 12 | 1100 | 28 | 11100 |
| 13 | 1101 | 29 | 11101 |
| 14 | 1110 | 30 | 11110 |
| 15 | 1111 | 31 | 11111 |

Now consider these three “8-light-bulbs” combinations in figure 1.5. Each one of these combinations of **on** and **off** light bulbs can be viewed as a **base-2** number.

**Figure 1.5**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

**01000001** (base-2) = **65** (base-10)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

**01000010** (base-2) = **66** (base-10)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

**01000011** (base-2) = **67** (base-10)

You are looking at **A**, **B**, **C** on the majority of today’s personal computers. By convention, at least the convention of the **A**merican **S**tandard of **C**omputer **I**nformation **I**nterchange (**ASCII)**, number **65** is used to store the letter **A**. Combinations **0** through **127** are used for a standard set of characters. The second group from **128** through **255** is used for a special extended set of characters.

We can use eight lights for each character that needs to be stored. All we have to do is place thousands of light bulbs in a container and you can store bunches of information by using this special **binary** code. Mathematically speaking, computations can be performed in any base. With the binary system, we have a means to store information and make electronic calculations possible as well.

We have learned that information can be stored in base-2 numbers. Base-2 numbers can store characters by using a system that equates numbers like the base-2 equivalent of **65** to **A**. At the same time, mathematical operations now become an electronic reality. In other words, the magic of on/off switches allows both electronic storing of information as well as electronic computation.

A single bulb can be **on** or **off** and this single light represents a single digit in **base-2**, called a **Bi**narydigi**t**, which is abbreviated to **Bit**. We also want to give a special name to the row of eight light bulbs (**Bit**s) that make up one character. This row shall be called a **Byte**. Keep in mind that **Byte** is not plural for **Bit**. There is one problem with storing characters in a single byte. You only have access to 256 different combinations or characters. This may be fine in the United States, but it is very inadequate for the international community. **Unicode** is now becoming very popular and this code stores characters in 2 bytes. The result is **65,536** different possible characters. Java has adopted **Unicode**, as have many technical organizations. The smaller **ASCII** code is a subset of **Unicode**.

|  |
| --- |
| **Bits, Bytes and Codes** |
| **Bit** is a **Bi**nary digi**t** that is either **0** (off) or **1** (on).  **1 Nibble = 4 bits**  **1 Byte = 8 bits.**  **1 Byte** has **256** different numerical combinations.  **2 Bytes** has **65,536** different numerical combinations.  **ASCII** uses one byte to store one character.  **Unicode** uses two bytes to store one character. |

Early computers did in fact use **one** vacuum tube for each **Bit**. Very large machines contained thousands of vacuum tubes with thousands of switches that could change the status of the tubes. Miles of wires connected different groups of vacuum tubes to organize the instructions that the computer had to follow. Early computer scientists had to walk inside giant computers and connect the wires to different parts of the computer to create a set of computer instructions.

The incredible advances in computer technology revolve around the size of the bit. In the forties, a bitwas a single vacuum tube that burned out very rapidly. Soon large vacuum tubes were replaced by smaller, more reliable, vacuum tubes. A pattern was set that would continue for decades. Small is not only smaller, it is also better. The small tube gave place to the pea-sized transistor, which was replaced by the planar transistor and bits were miniaturized, becoming smaller and smaller. Today, a mind-boggling quantity of bits fits on a microchip.

This is by no means a complete story of the workings of a computer. Very, very thick books exist that detail the precise job of every component of a computer. Computer hardware is a very complex topic that is constantly changing. Pick up a computer magazine, and you will be amazed by the new gadgets and the new computer terms that keep popping up. The intention of this brief introduction is to help you understand the essence of how a computer works. Everything revolves around the ability to process enormous quantities of binary code, which is capable of holding two different states: **1** and **0**.

**1.4 Memory and Secondary Storage**

Electronic appliances used to have complex --- cables everywhere --- dusty interiors. Repairing such appliances could be very time consuming. Appliances, computers included, still get dusty on the inside, but all the complex wires and vacuum tubes are gone. You will now see series of boards that all have hundreds and thousands of coppery lines crisscrossing everywhere. If one of these boards is bad, it is pulled out and replaced with a new board. What used to be loose, all over the place, vacuum tubes, transistors, resistors, capacitors and wires, is now neatly organized on one board. Electronic repair has become much faster and cheaper in the process.

In computers the main board with all the primary computer components is called the **motherboard**. Attached to the motherboard are important components that store and control information. These components are made out of chips of silicon. Silicon is a semiconductor, which allows precise control of the flow of electrons. Hence we have the names memory chip, processing chip, etc. We are primarily concerned with the **RAM** chip, **ROM** chip and the **CPU** chip.

It was mentioned earlier that information is stored in a binary code as a sequence of **ones** and **zeroes**. The manner in which this information is stored is not always the same. Suppose that you create a group of chips and control the **bits** on these chips in such a way that you cannot change their values. Every **bit** on the chip is fixed. Such a chip can have a permanent set of instructions encoded on it. These kinds of chips are found in cars, microwaves, cell phones and many electronic appliances that perform a similar task day after day.

Computers also have information chips that store permanent information. Such chips are called **R**ead **O**nly **M**emory chips or **ROM** chips. There is a bunch of information in the computer that should not disappear when the power is turned off, and this information should also not be altered if the computer programmer makes some mistake. A **ROM** chip can be compared to a music CD. You can listen to the music on the CD, but you cannot alter or erase any of the recordings.

Another type of chip stores information temporarily. Once again, information is stored in many **bytes**, each made up of eight **bits**, but this information requires a continuous electric current. When the power is gone, so is the information in these chips. Computer users also can alter the information of these chips when they use the computer. Such chips can store the data produced by using the computer, such as a research paper or it can store the current application being used by the computer. The name of this chip is **R**andom **A**ccess **M**emory chip or **RAM** chip.

The amount of memory in a computer is measured in bytes, not bits. Modern computers have so many bytes that usually memory is indicated as so many **Kilobytes** or **Megabytes** of memory. Kilobytes have already disappeared and MegaBytes are rapidly fading from the computer scene while **GigaBytes** and **TeraBytes** are alive and well in current computer terminology.

**Figure 1.6**

|  |  |  |
| --- | --- | --- |
| **Measuring Memory** | | |
| **KB** | Kilo Byte | 1 thousand bytes |
| **MB** | Mega Byte | 1 million bytes |
| **GB** | Giga Byte | 1 billion bytes |
| **TB** | Tera Byte | 1 trillion bytes |
| **PB** | Peta Byte | 1 quadrillion bytes |
| **EB** | Exa Byte | 1 quintillion bytes |

The *measuring memory* diagram, in figure 1.6, may get a frown or two. The information is technically incorrect. The diagram is meant to help remember the measure size in a rounded manner. After all, **Kilo** does mean one-thousand. Technically speaking one **KB** is **210** or **1,024** bytes. Using the same logic you can compute that one **MB**  is **220** or **1,048,576** bytes. I am sure you can live very comfortably using the previous “not really correct” diagram.

The most significant chunk of silicon in your computer is the **CPU** chip. **CPU** stands for **C**entral **P**rocessing **U**nit and this chip is the brain of the computer. You cannot call this chip **ROM** or **RAM**. On this tiny little chip are lots of permanent instructions that behave like **ROM**, and there are also many places where information is stored temporarily in the manner of a **RAM** chip. The **CPU** is one busy little chip. You name it, the CPU does the job.

A long list of operations could follow here but the key notion is that you understand that all the processing, calculating and information passing is controlled by the Central Processing Unit. The power of your computer, the capabilities of your computer, and the speed of your computer is based on your CPU chip more than any other computer component.

**Secondary Storage**

I just know that you are an alert student. **ROM** made good sense. **RAM** also made sense, but you are concerned. If the information in **RAM** is toast when you turn off the computer . . . then what happens to all the stored information, like your research paper? Oh, I underestimated your computer knowledge. You do know that we have external hard drives, CDs and USB jump drives that can store information permanently.

We have stored information on **rust** for quite some time. Did I say **rust**? Yes, I did and perhaps you feel more comfortable with **iron oxide**. Tiny particles of iron oxide on the surface of a tape or disk are magnetically charged positively or negatively. In a different manner than the internal computer, but with a similar logic, coded information is stored on a tape or a disk.

Please do keep in mind that this information will not disappear when the power is turned off, but it can be easily altered. New information can be stored over the previous information. A magnetic field of some type, like a library security gate, heat in a car, dust in a closet, and peanut butter in a lunch bag can do serious damage to your information.

You might be confused about the currently popular **CD-ROM**s. You can see that they are external to the computer, but **ROM** implies Read Only Memory. CDs store enormous amount of information. The information is permanent and thus behaves like **ROM**. When you use a CD with a computer it behaves as if you had added extra ROM to your computer internally. CDs do not use rust; they are far too sophisticated for such a crude process. The CD is coded with areas that reflect and absorb laser light. Once again we can create a code system because we have two different states, on and off.

The on/off state is the driving force of the digital computer. What is **digital**? Look at your watch. You can see digits, and you see the precise time. There is no fractional time. A clock with hour, minute and second hands is an **analog** device. It measures in a continuous fashion. A measuring tape is also **analog**, as is a speedometer with a rotating needle. What is the beauty of digitizing something? With digital information it is possible to always make a precise copy of the original.

It is easy to transfer, store and use digitized information. Entire pictures can be converted to a digitized file and used elsewhere. I am sure you have been in movie theaters where “digital” sound is advertised. So digital is the name of the game. Just remember that not all digitizing is equally fast.

The internal memory of the computer is digital and it uses electronics. The access of a hard disk involves electronics, but the information is read off a disk that rotates and only one small part of the disk is “readable” at one time. Accessing a disk drive is much slower than accessing internal memory.

**1.5 What Is Programming?**

Computer science is a highly complex field with many different branches of specialties. Traditionally, the introductory courses in computer science focus on programming. So what is programming? Let us start by straightening out some programming misconceptions. Frequently, I have heard the phrase: *just a second sir, let me finish programming the computer*. I decide to be quiet and not play teacher. The person “programming” the computer is using some type of data processing software. In offices everywhere, clerks are using computers for a wide variety of data processing needs. Now these clerks enter data, retrieve data, rearrange data, and sometimes do some very complex computer operations. However, in most cases they are not **programming** the computer. Touching a computer keyboard is not necessarily programming.

Think about the word **program**. At a concert, you are given a program. This concert program lists a sequence of performances. A university catalog includes a *program of studies,* which is a sequence of courses required for different college majors. You may hear the expression, *let us stick with our program,* which implies that people should stick to their agreed upon sequence of actions.

In every case, there seem to be two words said or implied: **sequence** and **actions**. There exist many programs all around us and in many cases the word program or programming is not used. A **recipe** is a program to cook something. A well- organized recipe will give precise quantities of ingredients, along with a sequence of instructions on how to use these ingredients.

Any parent who has ever purchased a *some-assembly-required* toy has had to wrestle with a sequence of instructions required to make the toy functional. So we should be able to summarize all this programming stuff, apply it to computers and place it in the definition diagram below.

|  |
| --- |
| **Program Definition** |
| A **program** is a sequence of instructions, which enables  a computer to perform a desired task.  A **programmer** is a person who writes a **program** for a computer |

Think of programming as communicating with somebody who has a very limited set of vocabulary. Also think that this person cannot handle any word that is mispronounced or misspelled. Furthermore, any attempt to include a new word, not in the known vocabulary, will fail. Your communication buddy cannot determine the meaning of a new word by the context of a sentence. Finally, it is not possible to use any type of sentence that has a special meaning, slang or otherwise. In other words, *kicking the bucket* means that some bucket somewhere receives a kick.

A very important point is made here. Students often think very logically, write a fine program, and only make some small error. Frequently, such students, it might be you or your friends, become frustrated and assume some lack of ability. It is far easier to accept that small errors will be made, and that the computer can only function with totally clear, unambiguous instructions. It is your job to learn this special type of communication.

**1.6 Program Languages**

You are happy with the fact that a program is a set of instructions to perform a given task. So what needs to be done is understand how instructions are communicated to a computer. At this stage, we have talked quite a bit about binary code, digital information and you have a pretty good hunch that our computers are not using *Star Trek* technology. So do not try to talk to a computer in English and tell it to finish your history homework.

For decades computer scientists have struggled with computers understanding human language. It has not been easy. We have made remarkable progress and a brief history of programming languages will help to understand where we are today, and what you will be doing this school year in computer science.

**Programming In Machine Code**

Programming the first computers was an unbelievably difficult task. Individual vacuum tubes had to be switched on or off. Instructions were sequenced in early computers by physically plugging wires from one computer memory segment to another. It is true that the early computers were amazingly fast. They were electronic, and they amazed people with their speed and accuracy, but programming those early computers was incredibly time consuming. Very few computers existed in those early days, and large teams of programmers and technicians were necessary for programming even simple jobs. Later, computers did improve the program process by allowing tape and cards to be used for program input. This was far more efficient than walking inside a computer, but there still remained the tedious process of thousands of **1s** and **0s** that had to be entered. Mistakes were very easily made, and very difficult to detect.

**Programming with Interpreters and Compilers**

If a program can perform many tasks then why not write a program that assists a person in writing a program. This is precisely the point made by the late **Grace Hopper**. Grace Hopper was a lieutenant in the Navy when she first worked extensively with computers. She was largely instrumental for developing translating programs that allow programming in a human-style language. She was one of the main developers of the popular program language, COBOL, and was very influential in the development of early translators. Two types of translating programs were created, **interpreters** and **compilers**. An interpreter takes a program and translates the program one line at a time. After each line is translated, the resulting machine code is executed. A compiler translates the entire program into a machine code file and then executes the file.

**Low-Level and High-Level Program Languages**

Languages that are very close to computer binary code are called **low-level**. Machine code and Assembly Language are low-level languages. Languages that are closer to human languages are called **high-level** languages. Some languages are very high-level today, and many programming tasks have already been performed. With many languages, it is possible to click and drag on program objects that have already been created and insert them inside your program.

So why not simply write your programs in English? Is that not a matter of creating some kind of translating program that takes English instructions and creates a machine code file for the computer? This is certainly what has been attempted for many years, but translating human languages has been very elusive. Consider the following example. In the sixties, computer scientists tried to write a program that would translate English into Russian and Russian into English. This is really the same problem of properly understanding the meaning of human language. The scientists decided to test their program by entering an English sentence. The resulting Russian sentence was then entered back into the computer and the final result should be the original English sentence. Should, that is, if the computer could translate correctly in both directions. The original sentence entered was:

*The spirit is willing but the flesh is weak*

I do not have a clue what the Russian result was, but I do know the final English result. Our computer scientists were quite surprised with

*The Vodka is great but the meat is rotten*

This little experiment showed the major problem with human languages. Human languages like English are idiomatic. We use all these idioms and proverbs, and slang and special nuances that are meaningless to the computers. How can computers figure out human language, when humans are confused? The bottom line is that programming requires a restricted language. This language can have human type words, but the words selected, the symbols and punctuation used all must have a very precise meaning.

The manner in which the program language structures its statements is called syntax. Program languages must have very precise syntax. Compilers first check to see if a program has correct syntax. Only after the syntax checks out, is the next step of translating into binary code performed.

**A Brief History of Program Languages**

The first successful programming language for the mathematics and scientific community, **FORTRAN** (**FOR**mula **TRAN**slation language), was released in 1956. FORTRAN was an excellent language for mathematics and science, but it could not handle the record processing required for the business world.

In 1960, **COBOL** (**CO**mmon **B**usiness **O**riented **L**anguage) was created (largely by Grace Hopper) for the business community and the armed forces. COBOL became extremely successful when the Department of Defense adopted COBOL as its official programming language.

**PL/1** (Programming **L**anguage 1) followed, trying to be a language for everybody. PL/1 attempted to be both an excellent language for science and for business. The result was an extremely cumbersome language that never gained much popularity. Then there were a host of many other short-lived languages.

**BASIC** (**B**eginner **A**ll-purpose **S**ymbolic **I**nstructional **C**ode) was designed for beginning college students. **BASIC** became the first popular program language for personal computers in the seventies.  **BASIC** required little memory, and it was the only language that could initially be handled by the first micro computers.

In the late seventies, early eighties, **Pascal** took a strong hold in the educational community. **Pascal** was developed by Niklaus Wirth, specifically for the purpose of teaching proper computer science programming techniques. This language was adopted for the Advanced Placement Computer Science Examination in 1984 and stayed in that position until the 1998 exam.

In the early seventies, the UNIX operating system was developed at the Bell laboratories. This operating system was written in several languages some of which were called **BCPL** or just plain **B**. A later version of the language was called **C**, since it followed **B**.

As the demands for sophisticated computer uses grew, so did the demand for ever more sophisticated computer programming languages. A new era with a powerful programming technique was born called *Object Oriented Programming* (**OOP**)*.* Bjarne Stroustrup combined the popularity of the existing C language with the demands for OOP and developed C.

C++ will continue to be a very important programming language for years to come. The industrial strength features and the tremendous number of programs written in C++ guarantee its survival.

**Java Comes on the Scene**

In the early Nineties Sun Microsystems worked on a language called **Oak**. The main focus of this new language was to be platform independent and object-oriented. Platform independent means that the language does not cause problems as programs are transported between different hardware and software platforms.

Oak was used internally by Sun Microsystems for about four years and released to the public in 1995 with the new name **Java**. Java was a big success largely because the new language was perfectly suited for the Internet.

Universities also adopted Java very rapidly. By the late Nineties object-oriented programming was taught everywhere and the new language Java gave no choice like C++. This appealed to many college professors who did not like the fact that C++ allowed, older non-OOP programming.

The College Board adopted Java as the AP Computer Science language to be used for its examination starting with the 2003-2004 school year. And you have just started a computer science course that will use Java as its programming language.

In 2009 Oracle purchased Sun Microsystems. Java has continued to improve in the same manner as when Sun Microsystems owned the company.

**1.7 Networking**

When you grow up in a certain environment, it can easily seem natural, as if the environment was always the way you see it. Today's students think it is perfectly normal that computers can use e-mail, play video games and surf the Internet. It was not always that simple. Computers evolved and yes computers do seem to evolve much faster than almost any other type of human creation.

**SneakerNet**

Early personal computers were not networked at all. Every computer was a stand-alone computer. Some computers were hooked up to printers and many others were not. If you needed to print something, and you were not directly connected to a printer, you stored your document on a floppy diskette, walked to a computer with an attached printer, and then printed the document. If a group of computer programmers worked together on a project, they needed to get up frequently with stored information to share it with other members in the team. Running around to share computer information is now called the *Sneaker* *Net* because sharing files or printing files requires you to *put on your sneakers* and walk to another computer.

**Peer-to-Peer Networks**

The first practical networks for personal computers were peer-to-peer networks. A peer-to-peer network is a small group of computers with a common purpose all connected to each other. This small network allowed a single printer to be used by any computer on the network and computers could also share information. These types of networks were called *Local Area Networks* or LANs. Initially, the networks were true *peer-to-peer* networks. This means that every computer on the network was equal. All computers were personal computer work stations.

**Client-Server Networks**

Peer-to-peer networks do not work well when networks get large. Special, dedicated computers, called servers, were needed. A server is a specialty computer that is connected to the LAN for one or more purposes. Servers can be used for printing, logon authentications, permanent data storage, web site management and communication. Many businesses would have multiple servers set up in such a manner that some servers exist for the purpose of backing up the primary server. Using backup systems, tape storage or other backup means insured computer reliability in case of computer failure.

**The Department Of Defense Networking Role**

It may come as a shock to you, but the Internet was not created so that teenagers could play video games and download music. The Internet has its origins in the "Cold War." If you do not know what the "Cold War" is ask your Social Studies teacher, your parents or any old person over 30. During the Cold War there was a major concern about the country being paralyzed by a direct nuclear hit on the Pentagon. It used to be that all military communications traveled through the Pentagon. If some enemy force could knock out the Pentagon, the rest of the military establishment communication would be lost. A means of communication had to be created that was capable to keep working regardless of damage created anywhere. This was the birth of the Internet. The Internet has no central location where all the control computers are located. Any part of the Internet can be damaged and all information will then travel around the damaged area.

**The Modern Internet**

Students often think that the Internet is free. Well that would be lovely, but billions of dollars are invested in networking equipment that needs to be paid in some fashion. Computers all over the world are first connected to computers within their own business or school. Normally, businesses and schools have a series of LANs that all connect into a large network called an *Intranet*. An *Intranet* behaves like the Internet on a local business level. This promotes security, speed and saves money.

Now the moment a school, a business, your home, wants to be connected to the outside world and the giant world-wide network known as the Internet, you have access to millions of lines of telecommunications. This will cost money and every person, every school and every business who wants this access needs to use an *Internet Service Provider* or ISP. You pay a monthly fee to the ISP for the Internet connection. The amount of money you pay depends on the amount of traffic that flows through your connection and the speed of your Internet connection.

Today many computers use a wireless connection to hook up to some local network, that in turn hooks up to the Internet. Wireless connections are convenient, but there are some problems. Signals are not always reliable, just like cell phones. You may be stuck in an area somewhere where the signal is weak. Furthermore, there is the security issue. Information that travels wireless is much easier to pick up by hackers than information that is channeled through cable.

**1.8 Hardware and Software**

Computer science, like all technical fields, has a huge library of technical terms and acronyms. Volumes can be filled with all kinds of technical vocabulary. Have no fear; you will not be exposed to volumes, but at the end of this chapter you do need some exposure to the more common terms you will encounter in the computer world. You have already learned about different types of memory, programming languages and networking. There are a few important computer terms that you need to understand.

For starters, it is important that you understand the hardware and software difference. These are the two biggest divisions in the computer business. Hardware involves all the computer components that can be seen, felt, picked up and dropped, like a monitor, a mouse, a jump drive. Software involves the set of computer instructions that are coded on an USB jump drive, a CD, or a hard drive.

**Computer Hardware and Peripheral Devices**

There are big, visible hardware items that most students know because such items are difficult to miss. This type of hardware includes the main computer box, the monitor, printer, and scanner. There are additional hardware items that are not quite as easy to detect.

It helps to start at the most essential computer components. There is the CPU (Central Processing Unit), which controls the computer operations. The CPU together with the primary memory storage represents the actual computer. Frequently, when people say to move the CPU to some desk, they mean the big box that contains the CPU and computer memory. In reality it contains lots more than just a bunch of memory chips. There are also many peripheral devices.

What does periphery mean? It means an imprecise boundary. If the computers are located on the periphery of the classroom, then the computers are located against the walls of the classroom. Computer hardware falls into two categories. There are internal peripheral devices and external peripheral devices.

External peripheral devices were already mentioned. These are hardware, located outside the computer and connected with some interface, which is usually a cable, but it can also be wireless. The first external peripheral device you see is the monitor. In the old days a monitor was called a CRT (Cathode Ray Tube). This was appropriate with the bulky monitors that looked like old televisions. Today many monitors use LCD (Liquid Crystal Display) or Plasma screens. It is common now for monitors to be 17, 24, or even 28 inches. (Right now, I am actually looking at a 48 inch plasma screen as I edit the 2012 version of this chapter.) Things have has changed considerably since the days of 10 inch computer monitors.

Other external peripheral devices include a printer, keyboard, mouse, scanner, jump drive or other memory stick.

There are many internal peripheral devices that are connected to the computer inside the computer case. These devices include the disk drive, CD ROM drive, digital camera memory drive, network interface card and video card.

**Computer Software**

Computer software provides instructions to a computer. The most important aspect of this course is to learn how to give correct and logical instructions to a computer with the help of a programming language.

Software falls into two categories. There is system software and application software. Usually, students entering high school are already familiar with applications software.

**Applications software** runs an application on a computer. Cell phones call this an *app*. The whole reason why a computer exists is so that it can assist people in some type of application. For centuries, accounting kept large, complicated spreadsheets of business numbers. Now these same numbers are entered on an electronic spreadsheet. You will not experience the horror of finding some mistakes on the early pages of a thirty page, typed, research paper. This type of mistake necessitates retyping the majority of the pages. Today, this is a simple matter of inserting the corrections on a word processor program. Spread sheets and word processors were the first software applications. Currently, there are thousands of other applications to assist people in every possible area from completing tax returns to playing video games.

**System software** involves the instructions that the computer requires to operate properly. A common term is OS (operating system). The major operating systems are Windows, UNIX, Linux and the MAC OS. It is important that you understand the operation of your operating system. With an OS you can store, move and organize data. You can install new external devices like printers and scanners. You can personalize your computer with a desktop appearance and color selections. You can install additional applications and computer protection against losing data and viruses.

**1.9 Summary**

This has been an introductory hodge-podge chapter. It is awkward to jump straight into computer science without any type of introduction. Students arrive at a first computer science course with a wide variety of technology backgrounds. Some students know a little keyboarding and Internet access along with basic word processing skills taught in earlier grades. Other students come to computer science with a sophisticated degree of knowledge that can include a thorough understanding of operating systems and frequently knowledge of one or more program languages as well.

The secret of computer information storage and calculation is the binary system. Information is stored in a computer with combinations of base-2 **ones** and **zeroes**. Individual binary digits (**bits)** store a one or a zero. A one means true and a zero means false. A set of eight bits forms one **byte**.

A byte can store one character in memory with **ASCII**, which allows **256** different characters. The newer, international **Unicode** stores one character in two bytes for a total of **65536** different characters.

Neither this chapter nor this book explained any details about the operating system. Operating systems change frequently, or at least operating system versions change to a new-and-improved model about every two or three years. A solid knowledge of your computer's operating system is vital. Writing a computer program requires knowledge of editing text, saving and loading files and moving around the hard drive's directory system efficiently.

If your basic computer knowledge is weak, make sure to pick up additional information from your teacher, library or bookstore. Technology is evolving and students arrive in computer science classes with increased sophistication. In this course no prior knowledge whatsoever about any programming language or programming logic is assumed. However, a fundamental knowledge of basic computer operations is essential.

Sun Microsystems created Java to be a programming language that is portable on many computer platforms, a so-called *platform-independent* language. They also wanted the language to be compatible with web page development.

Early computers were stand-alone work stations. The first networked computers used a "peer-to-peer" network. This was followed by LANs (Local Area Networks) that connected dedicated specialty servers with computers and printers for a common purpose. The Department of Defense developed the Internet as a means to provide communication at war time.

Individuals, schools and businesses can set up a LAN at their private location without paying a fee beyond the cost of the necessary hardware and software. Connection to the Internet requires an ISP (Internet Service Provider) and a monthly connection fee.

Many computers today, especially laptop computers, have wireless network connections. Such connections are convenient, but they are not as reliable and there is also a greater security risk.

Computers use hardware and software. Hardware peripheral devices are the visible computer components. There are external peripheral devices, such as monitors, keyboards, printers and scanners. There are also internal peripheral devices like disk drives, CD ROM drives, network interface cards and video cards.

Software falls into two categories of application software and operating system software. Application software includes the common applications of word processing and spreadsheets, but also tax return software and video games. Operating system software runs the computer and allows the user to personalize the computer to his or her needs and organize data.