"It Went 'BOING' !"

by

Andy Johnson

version 1.3

Textbook for
CSC101 - Introduction to Computers
Wayne State University
Detroit Michigan.

Disclaimer: The author disclaims any damages caused by the use or misuse of information contained in this text.

Cover art by Gary Thomas Washington.

Lots of thanks to Jason Leigh, who deserves half the credit (or blame) for the ideas expressed in this text.

I would also like to thank Preston Nevins and Raymond Pelzer for their help in reviewing this text, and Wayne Brillhart for some of his thoughts that found a home here.

Contents

List of Figures

.

0.1 Preface

Every 'real' textbook has a section called "the Preface." Of course no one actually reads the preface except for those people who helped the author while he was writing it. They read the preface to see if their name is mentioned in the acknowledgements.

Since you took the time to read this, I'll explain why you are holding a course-pak in your hands and not a 'real' textbook. To do this we must go back a few years to the summer of 1989. OK, so that's not so far back. Dinosaurs didn't roam freely on the planet surface then, and Ike wasn't in the White House, and bell-bottoms weren't the fashion of the day ... but its still a few years ago. The summer of 1989 was the first time that I taught CSC101 at Wayne State University. There were two instructors for this particular class: Jason Leigh and myself - See Figure 0.1. It was summer and we were quite bored. We decided to teach this class a little differently. We came up with a technique which we called 'stereo teaching.' Basically, we both showed up for lecture and we both taught the class, though usually not at the same time. This kept the students awake, kept us awake, and made for some rather entertaining classes.

We quickly found that the students were not satisfied with the textbook and workbook that we were using. Since we could not find a better one, we gathered our handouts together and created our own workbook for the Fall Term of 1989. By the Summer Term of 1990 I compiled our lectures into the first version of this textbook and made it available to the students. Both books have gone through several revisions since then as students made comments and suggestions.

CSC101 at Wayne State University serves two purposes. It is the first class in the Computer Science Curriculum as well as one of the primary was for Liberal Arts students to satisfy the university's computer literacy requirement. Students come into this class, and begin reading this text, knowing nothing about computers. By the end of the class most are confidently using computers.

Having taken more university classes than I care to remember, and having suffered through many truly dreadful textbooks; I tried to write this text for the student as

Figure 0.1: Andy Johnson and Jason Leigh

well as the instructor. This book 'sounds' more like a lecture than it 'reads' like a textbook, and I think you will find it entertaining. I hope you enjoy it.

Chapter 1

What is a Computer?

It went *zip* when it moved,

and *ptht* when it stopped,

and *whrr* when it stood still.

I never knew just what it was,

and I guess I never will.
– Peter, Paul and Mary.

Computers have always enjoyed a certain popularity in the movies. From the "electronic brains" of the 50's with the technicians in white lab coats turning dials and pulling big iron levers, through the blinky-light panels and punch-card readers of the 60's that would spark and spew out smoke at the slightest trouble. In the 70's theatrical computers moved beyond blinking lights to swirling colour patterns where technicians wore bell-bottoms and instructed their machines in English. Invariably, all these machines tried to take over the world and would do very well until the hero asked them to calculate the value of π. The computer would then explode in a blaze of cheap special effects. In real life computers are about as exciting as a toaster, and

are becoming nearly as common. You don't need a fancy degree or a white lab coat to work with one, and they are usually used for much more mundane purposes than conquering the world. Very simply, a computer is a device used to store, process, and retrieve information.

A computer is a tool. As human civilization has advanced we have created the tools that we needed to survive and prosper. As times change, our needs change, and the tools we require change. New tools are invented, improved, and eventually discarded. Instead of weaving textiles, or forging metals, we are now manipulating information.

1.1 Information Age

When computers were first being built during World War II,there was a belief that only a handful would be needed around the world. At the time it was true. But information has become more and more important to our society. We are now in the midst of the **information age** - where information itself is an important raw material to be processed. There is now a vast amount of raw information to sift through, calculate with, and transform into a finished product. Computers are well suited to this task.

Today computers are everywhere. Banks use computers to run their ATM machines. Grocery stores use them to price your food. Wayne State uses them to register you for classes. Other students are using them to type up term papers and do calculations for their lab reports. Learning to use a computer can make your life as a student much easier, but taking those initial first steps are difficult. Computers can be imposing. This class is designed to help you take those first steps and acquire the skills to use a computer productively.

Computers come in a variety of shapes and sizes, perform a variety of features, and have a correspondingly wide range in the sticker price. Different computers have strengths in different areas so there is no such thing as a 'best' computer. Unfortunately there are a lot of people out there who earn their living by telling you that there is a 'best' computer, and that their company builds it. The 'best' computer depends on the jobs that you need the machine to perform, how easily you want to perform them, and the amount of money you are willing to spend. The quality of a particular machine depends on the person using it. Like automobile manufacturers, computer manufacturers tend to release different models of their machines depending on the features the user needs. Ford manufactures automobiles with 2 doors, 3 doors,

and 4 doors. They manufacture sports cars and delivery vans. So does GM and Toyota. There is no 'best' car, but there can be a 'best' car for a given job, and price range.

1.2 Personal Computer

When we talk about a computer in this class, we will usually be talking about a **Personal Computer**. Large businesses and government agencies have been using computers for 40 years, but it is only within the last decade that the 'average person' has gotten access to this powerful tool. It was not until the mid to late 70's that the technology became available to the people in boxes with names like the TRS-80 Model I, and the Apple][. Of course, at the time, people didn't know what to do with them. Personal computers were like ham radios - neat for tinkering with, and a diverting hobby. But now we have entered the 90's, and personal computers are becoming as common as microwave ovens and VCRs.

As an undergraduate, most of the computers that you will encounter on campus are personal computers. A personal computer will sit comfortably on a desk, and is used by a single person (which is why it's called a personal computer.) There are many popular brands of personal computers being produced, and you will find several different types on campus: IBM-PCs, Apple Macintoshes, and Commodore Amigas to name a few. While these machines may look slightly different, they all operate on the same basic principles, and have the same basic parts. Once you have learned to use one personal computer, you have learned about all computers.

Of course, this may not sound very 'personal,' and you may not want to get on intimate terms with a computer. But some of them can be quite friendly and patient. Douglas Adams came up with the most 'personal' personal computer in *the Hitchhiker's Guide to the Galaxy* and described it as:

> ...a device that looked like a largish electronic calculator. This had about a hundred tiny flat press buttons and a screen about four inches square on which any one of a million "pages" could be summoned at a moment's notice. It looked insanely complicated, and this was one of the reasons why the snug plastic cover it fitted into had the words DON'T PANIC printed on it in large friendly letters.

Now the phrase "Don't Panic" will be very useful in this class, even though it is not written in large friendly letters on any of the major computer brands. We are

going to cover a lot of material, and you are going to see a lot of new terms. You may feel rather confused at times (as the student saying goes - "speed kills!") but computing has its own language, and you need to know the lingo. A good way to unconfuse yourself is to sit down and work with a personal computer. Let the words become objects, and actions in the real world. Reading about how to drive a car is very confusing until you get out there on the road ...and then it all seems to make sense ...either that, or you drive off a bridge. That's one advantage you have learning about computers ...the computer labs have a very low mortality rate.

That brings us to a topic that is not often addressed ...fear. We are afraid of the unknown. We are afraid of doing something wrong. We might get hurt, or even worse ...we might look foolish. Computers are scary. They sit smugly on the desk waiting for you to make the slightest mistake. You touch a key and then "whoop whoop whoop" ...sirens, alarms, bells, and everyone turns around to stare at you. Then you start babbling like an idiot, trying to explain yourself to the heavily armed "computer police" as the computer chalks up another victory. Now think about this. Would a company that builds a computer like that actually stay in business for long? No. Computer companies do not want to scare off prospective customers. They want customers that will be happy and feel comfortable with their computer. Would a computer company build a personal computer that could only be used by college graduates? No. That would tend to limit sales as well. Would a computer company build a personal computer that falls apart at the slightest touch? No. That would tend to decrease customer satisfaction. Would a computer company build a personal computer that explodes when you touch it? No. That would be very bad for business. So there really isn't anything to fear, except perhaps that you will find the machine so useful that you can't live without it.

So what really is a computer? Computers are made up of two main parts. One is **hardware** - the physical parts of the computer. The other is **software** - the instructions that control the hardware. Its customary in texts like this to use a lot of similes and say "a computer is like this," and "a computer is like that." I would hate to break with tradition at this time, so I will say that a computer is like a stereo system. A stereo system is made up of components such as a tuner, an amplifier, a CD player or turntable, a cassette deck, and some speakers. These are hardware. They are physical. You can touch them. A compact disc and a cassette tape are also pieces of hardware, and they contain music on them. This music is the software. Without the software, the hardware can't do anything productive. Without the hardware, the software can't do anything productive. I started out by saying a computer is a device to store, process, and retrieve information. A stereo system is a device used to

store, process, and retrieve information in the form of music. A stereo stores music on cassette tape. It retrieves music from a cassette or CD. It probably has a volume control, and possibly a graphic equalizer to process the music.

1.3 Hardware

Computers are also made up of different components that are connected together. I'm sure you've seen stereo systems where the amplifier is built in with the tuner, or the cassette deck. Different brands of stereos look different and have different types of controls. The same is true of computers. Most modern personal computers have the following parts: **Main Unit**, **Disc Drive**, **Monitor**, **Keyboard**, and a **Mouse**. The main unit holds the guts of the computer ... its like the amplifier. The disc drive is used to store information for long periods of time ... its like the cassette deck. The monitor is just a fancy name for the thing that looks like a TV screen. It is where we get the main **output** from the computer. It is like the speakers. The keyboard and mouse allow us to give commands to the computer ... the **input**. They're like the buttons and dials on the components.

1.4 Software

Computers require software to run. Software supplies the instructions for the computer to follow. The most important piece of software a computer has is called its **operating system**. This is software that allows the computer to go about its business (it allows the system to operate) and lets it talk to all of its components. The operating system usually comes with the computer, but you will soon need other software. Just like your stereo can be used to play different kinds of music, a computer can run various different types of software. A Word Processor turns your computer into a very fancy typewriter. A Spreadsheet turns your computer into a very fancy calculator. A Database turns your computer into a very fancy rolodex.

1.5 Personal Computers and You

The first popular personal computers of the mid 70's were very primitive by today's standards. These had simple keyboards and black& white screens that could only display letters or numbers. The software was stored on cassette tapes. These machines

were very slow, very expensive, and very hard to operate by today's standards. Within the last 15 years personal computers have become much more powerful, much faster, much easier to use ...and the price has actually stayed about the same. One of the main reasons for this is that more people have found ways to use personal computers to help them with their work. Personal computers moved beyond hobbyists and into the mainstream. They moved out of the basement, put on a dull grey suit, and headed off into the business world.

The question you need to answer is "Why do I need to learn how to use a personal computer?" One answer comes from Arno Penzias who wrote in his book *Ideas and Information* "Success usually comes to those who apply technology to their best advantage." My answer is "Because its interesting, and because its fun."

1.6 Questions

1. What is the difference between hardware and software?

2. Which of your daily activities are affected by computers?

Chapter 2

Hardware

When you go to sit down in front of a computer, it is a good idea to know what each of the parts is called. This way you can sound cool if you need to ask for help. It is much 'cooler' and much more helpful to say "my floppy disc will not initialize" rather than "it doesn't work." So we are going to take a look at each of the components of a computer. Figure 2.1 shows a typical personal computer.

2.1 Monitor

The monitor is the part of a computer that looks like a TV screen. The monitor has a **screen** just like a TV does. It is used to display visual information to the user. Computer monitors usually sit on top of the **main unit** of the computer, though most can be positioned wherever the user pleases. Monitors look a lot like TV sets on the inside as well. All use the same "cathode ray-tube" technology that TVs use. Like TV sets, monitors tend to get hot so they have ventilation slots on top to let heat out. Its a good idea to keep these slots clear (i.e., don't put your papers on top of the monitor.)

Some monitors are colour and some are monochrome (single coloured.) Monochrome monitors are usually coloured either green (because that's how computers are supposed to look) or amber (because people were going blind staring at their green screens.) RGB monitors are high-grade colour monitors. **RGB** stands for Red Green Blue (the three primary colors for light), and the intensity of each color can be set independently for each pixel on the screen. A **pixel** is a picture element (or 'dot' on the screen.) Pixels are lit up in patterns to form characters and pictures. If you stare

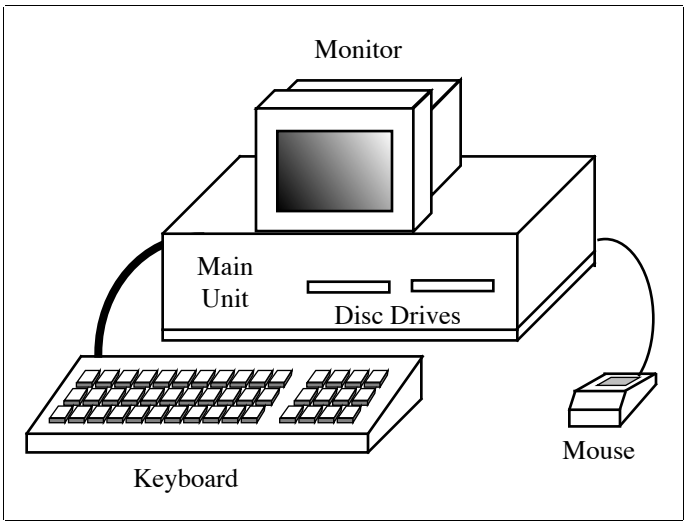


Figure 2.1: A Typical Personal Computer

real close at the monitor screen (and ruin your eyes) you can see the individual pixels. A typical computer monitor is 640 pixels wide by 480 pixels tall, meaning there is a total of 307,200 pixels on the screen. These monitors usually have a **resolution** of 72 pixels per inch. Some computers allow you to use your TV as a computer monitor, though the picture is not as sharp as a real monitor since TVs do not have as good a resolution as monitors do.

Today the monitor is the biggest, heaviest, and most power draining part of a personal computer system. This should change within five years as low power **flat screen** displays become better, and replace the current CRT technology. As the name suggests, flat screens are flat. They do not use the bulky picture tube of current televisions, but more exotic technologies such as liquid crystal or gas-plasma displays. Currently these types of displays are in use with lap-top computers. Monochrome flat screen displays are now quite cheap, and colour flat screen displays are expensive but available.

When you were a kid your mother probably yelled at you not to sit so close to the TV set because it would ruin your eyes. Well, moms are always rather paranoid, but in this case she might have been right. The electromagnetic emissions from CRTs may be dangerous. Ever since the mid 70's this has been debated, and we will probably not know for sure for several more years. Just to be safe you should sit a discrete distance away from the monitor. Keeping your head at least 2 - 3 feet away from the

screen is a good idea.

2.2 Speaker

Since you have video, it seems obvious that you should also have audio. All personal computers have a built in speaker, and most have multi- voice stereo sound, along with the ability to play the sound through an amplifier.

2.3 Printer

If you wrote your term paper on a computer, it would be very inconvenient for you to bring your computer to your Professor so he could read the monitor screen. You may need to get a **hard copy** or **printout** of your work onto paper. A printer is a device that does this. It prints characters, numbers, or graphics onto paper. Printers do not come with the computer, but are a necessary addition. There are several different types of printers available including (in ascending order of spiffiness) letter-quality, dot-matrix, ink-jet and laser printers. Figure 2.3 gives a summary.

2.3.1 Letter Quality Printers

Letter-Quality printers print just like typewriters do with a special print head for each character. They print text as well as a typewriter, but can not do graphics or special characters. While these printers were popular in the early 80's, their inability to print graphics or special characters is a very big limitation today. They usually cost around $150.

2.3.2 Dot Matrix Printers

The most common type of printer is a dot matrix printer. Unlike a typewriter which has an individual print head for each character, a dot matrix printer has a single print head made up of a matrix (or array) of pins. Depending on what is to be printed, different pins are pressed into the ribbon creating a pattern of dots on the paper. The pins can be set to represent a letter, or print out graphics. Common dot matrix printers are 144 **DPI** (dots per inch). This is twice as many as the monitor, but not enough to give really good output as the individual dots are quite visible. The difference between dot matrix and letter quality letters is shown in Figure 2.2 Colour

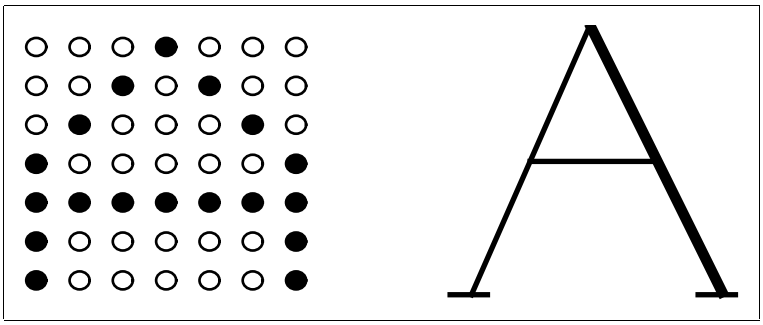


Figure 2.2: Dot Matrix 'A' VS Letter Quality 'A'

ribbons are now becoming quite popular allowing dot matrix printers to produce multi-colour printouts. Dot matrix printers are very good for general purpose work, and are fairly inexpensive - around $300.

2.3.3 Ink-jet Printers

Ink-Jet printers do not have print heads. Instead they direct a stream of ink at the page. This allows them to print very fine text or graphics. These are typically 200 dots per inch, so they are better than dot matrix printers for general work. They also cost slightly more - $900.

2.3.4 Laser Printers

Dot-Matrix, letter quality, and ink-jet printers print one line at a time. Laser Printers are the newest form of printer and they print one page at a time like a xerox machine. Laser Printers have their own computer on board to direct the laser which "draws" on a photosensitive drum giving the drum a static charge. Particles from a toner cartridge are attracted to the drum which is then pressed onto the paper. Technically they are still dot matrix printers but they have a resolution of 300 dots per inch, so the characters are almost indistinguishable from letter-quality. Since laser printers are dot matrix, they can print graphics as well as special characters with extreme precision. Laser printers are very advanced and so they cost quite a bit - $1500. Color laser printers can cost $5000.

Printer Type	Text	Graphics	DPI	Cost
Letter-Quality	Y	N	n/a	$150
Dot Matrix	Y	Y	144	$300
Ink-Jet	Y	Y	200	$900
Laser	Y	Y	300	$1500

Figure 2.3: Comparison of Printer Types

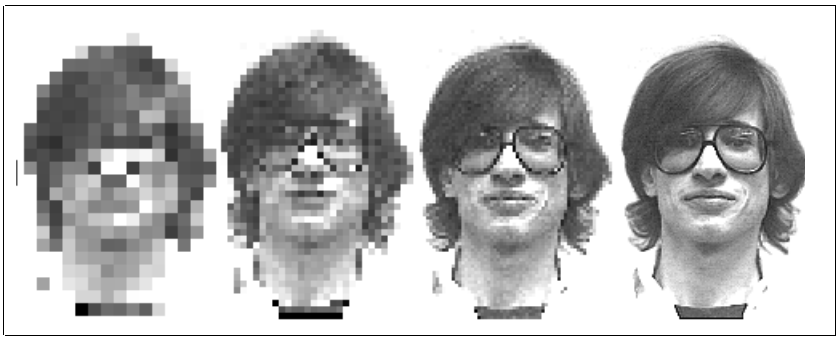


Figure 2.4: 9 DPI 18DPI 36DPI 72DPI

2.3.5 Resolution

The quality of both monitors and printers is partially measured in terms of their resolution. A monitor has a certain number of pixels per inch, and a dot-matrix printer can print a certain number of dots per inch. To show you how important resolution is, Figure 2.4 shows the same image viewed with an increasing degree of resolution.

2.4 Keyboard

The keyboard of a computer looks a lot like the typical "QWERTY" keyboard of a typewriter, except with a few extra keys around the sides. The keyboard allows the user to type commands, or a term paper into the computer. Most keyboards are **detachable**. Detachable keyboards are connected to the main unit by a cable so you can pull the keyboard away from the main unit and use it in a more relaxed position.

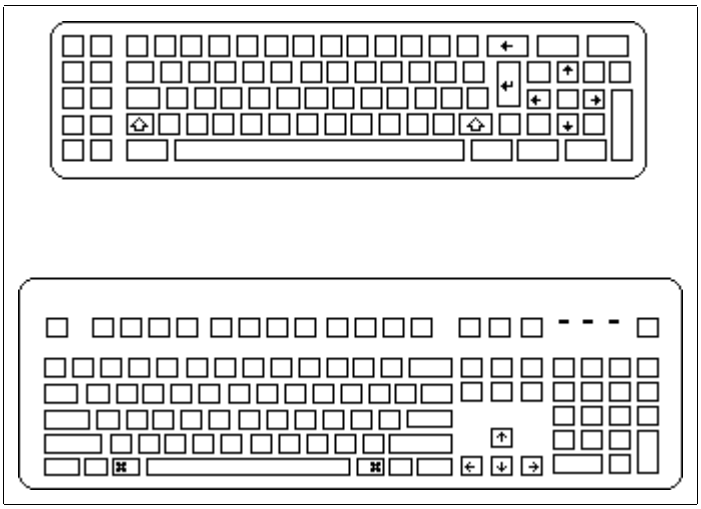


Figure 2.5: IBM-PC(top), and Mac II(bottom) keyboards

Larger keyboards as shown in Figure 2.5 come with function keys, arrow keys, and numeric keypads as well as the standard typewriter keys.

Function keys are aptly named as each of these keys is set so that pressing it causes some function to be performed, rather than some character to appear. **Arrow keys** allow the user to move up, down, left, and right around the screen. **Numeric keypads** put the number keys, and keys needed for simple arithmetic into a convenient pattern. Standard typewriter keyboards have the numbers across the top in a row which is very inconvenient if you need to type in a lot of numbers. Now you might expect that the numeric keypad arranges the number keys in the most intuitive pattern. It doesn't. The phone company found the most intuitive way of placing the numeric keys, and patented it. This left the computer companies to use the second most intuitive way of placing the keys.

2.5 Mouse

A mouse looks very little like its mammalian namesake. It doesn't have a face, or legs, or fur, but it does have a tail. It looks much more like a bar of soap with one or more buttons on top, and a ball on the bottom. One assumes that the person who named the little critter had been working just a little too hard that day. Mice are used for drawing and making selections from alternatives displayed on the monitor screen, by

moving the beast around and clicking the button(s). The number of buttons on the top of the mouse varies with the brand of computer. Some mice have one button (such as the Macintosh,) some have two buttons (such as the Amiga, and the IBM-PS/2) and some have three buttons (such as the Apollo, and the Sun.) One mouse has been released with 40 buttons on it. Since mice are relatively new, there has not been enough research done to really say how many buttons is "best", so different companies have chosen to use different numbers of buttons. The question is not likely to be resolved soon. After all, automobile manufactures still can't agree where to put reverse in a manual transmission.

2.6 Main Unit

The monitor, printer, keyboard, and mouse are all attached by cables to the Main Unit of the computer. This is the expensive box that contains the guts of the machine. Most importantly, this box contains the CPU, the RAM, and the ROM. Since the computer does a lot of heavy thinking in here, the main unit gets quite hot. Most have a fan inside to keep air circulating around the box.

2.6.1 Motherboard

When we open up the main unit (and possibly void the warranty on our machine) we see one large green **printed circuit board** with **chips**, and possibly some **cards**, plugged in to it. The main board is called the motherboard. This is where the actual computing is done by the machine. Looking down on a motherboard one is reminded of looking down on a small city from an airplane. The chips are arranged in neat rows, like buildings on grassy streets. Information in the form of electrical impulses moves through these "streets," into, and out of the various chips like so many little delivery vans.

2.6.2 CPU

In the center of our grassy town lies one big important building - the CPU. CPU stands for "Central Processing Unit." The CPU is what controls the operation of the computer. In the 50's terminology it would have been called the "electronic brain." The two most common CPUs today are made by Intel Corp. and Motorolla Corp. Intel makes CPUs for IBM machines and gives them wonderfully descriptive

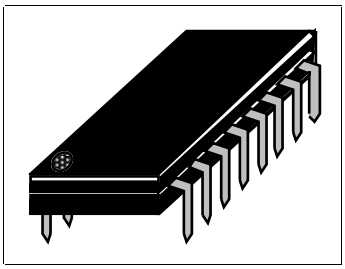


Figure 2.6: A Chip

names like 8088 ("eighty eightyeight"), 80286 ("eighty two eightysix"), 80386, and 80486. Motorolla makes CPUs for machines like the Macintosh and gives them equally memorable names like 68000 ("sixtyeight thousand"), 68020 ("sixtyeight thousand twenty"), and 68030.

CPUs are rated by how fast they can do their work. There are several ways to measure the speed of a CPU chip, but most of them are useless since it is not just the brand of CPU that determines the speed of a computer. However, when you look at ads for computers they will usually say the computer runs at 8 Mhz or 15 Mhz or 40 Mhz. This is the **clock rate** of the CPU. The larger the number of Megahertz, the faster the machine should be able to do its work. Other companies rate their machines in some number of MIPS, meaning " Millions of Instructions Per Second." A more realistic definition of MIPS is " Meaningless Indication of Processor Speed."

2.6.3 Chip

The CPU is a chip. Chip is short for **DIP chip**, and DIP stands for "Dual In-Line Package." These are the things that look like boxy black centipedes. They have a thin rectangular body, and two rows of legs on their sides (i.e. each chip is a package with a "dual" set of legs that are "inline.") This strange insect has a plastic body, metal legs and silicon innards. The actual guts of the chip is a very small wafer of silicon, smaller than your fingernail. Something this small is hard to work with, so the silicon wafer is encased in this large plastic body. The metal legs connect the chip to the motherboard so electricity can flow into the wafer. A chip is shown in Figure 2.6.

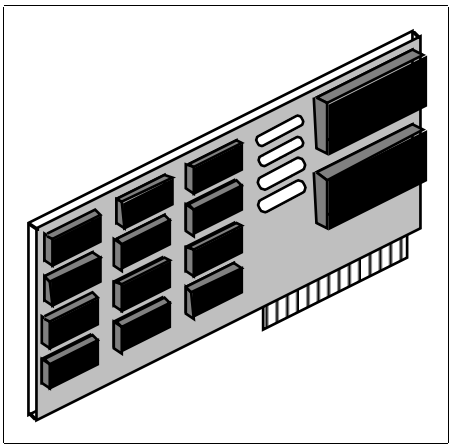


Figure 2.7: A Card

2.6.4 Card

Most motherboards have slots in them. These slots allow you to plug cards into the motherboard. A card is another printed circuit board containing chips which expand the capabilities of your machine. Expandability is a good feature to look for in a computer. By the time you buy a personal computer it is already years out of date. There is always something new and better coming out, and its nice to be able to plug some of these new and better things into your machine instead of buying a whole new computer every six months. A card is shown in Figure 2.7.

2.7 0s and 1s

At this point we need to have a brief but painful discussion as to how computers actually work. Computers run on electricity. A computer can be thought of as a large number of switches that are either set to on or off. Because of this, computers work in this world of 0s (off) and 1s (on). That is all that they know. All the CPU does is work with these 0s and 1s. Combining strings of 0s and 1s together allows computers to manipulate a vast amount of information. For example: Morse Code conveys a lot of information by stringing together 0s and 1s in the form of dots and dashes as shown in Figure 2.8.

Morse code was designed so the more commonly used letters have fewer 0s and 1s than less frequently used letters so the overall transmission speed is faster. With

a=	● -	j=	● - - -	s=	● ● ●
b=	- ● ● ●	k=	- ● -	t=	-
c=	- ● - ●	l=	● - ● ●	u=	● ● -
d=	- ● ●	m=	- -	v=	● ● ● -
e=	●	n=	- ●	w=	● - -
f=	● ● - ●	o=	- - -	x=	- ● ● -
g=	- - ●	p=	● - - ●	y=	- ● - -
h=	● ● ● ●	q=	- - ● -	z=	- - ● ●
i=	● ●	r=	● - ●		

Figure 2.8: Morse Code

a computer, the number of 0s and 1s for all the typeable characters is 8, so we can arrange the letters in alphabetical order. One such standard ordering is called ASCII ("askey." short for American Standard Code for Information Interchange) Figure 2.9 shows only the lower case letters. The full ASCII table has 128 entries (from 00000000 to 01111111) containing upper-case letters, lower case letters, numbers, punctuation, and more.

2.8 Bits & Bytes

In a computer, a single 0 or 1 is called a bit. Bit is short for Binary digIT. Like inches, bits are a little small if we want to measure something large; so we can convert bits into larger units of measure. A string of 8 bits (such as 01100101) is called 1 byte. In order to represent a character such as "e" the computer needs 1 byte. When we measure the length of a paper, we do not count letters. We need larger units of measure such as words, or pages. A string of 1024 bytes is called 1 kilobyte (which you would think means 1000 bytes, but computer science people are a little weird so 1 kilobyte is 1024 bytes.) 1024 K (or 1024 x 1024 bytes) is called 1 M (megabyte.) As the technology improves every year, larger units of measure become commonplace, and new even-larger units of measure are then needed. Figure 2.10 summarizes.

So lets do a sample conversion. Lets say we want to find out how many bits are in 20M. Here's how we do it:

a=	01100001	j=	01101010	s=	01110011
b=	01100010	k=	01101011	t=	01110100
c=	01100011	l=	01101100	u=	01110101
d=	01100100	m=	01101101	v=	01110110
e=	01100101	n=	01101110	w=	01110111
f=	01100110	o=	01101111	x=	01111000
g=	01100111	p=	01110000	y=	01111001
h=	01101000	q=	01110001	z=	01111010
i=	01101001	r=	01110010		

Figure 2.9: ASCII

1 bit	=	0 or 1
8 bits	=	1 byte
1024 bytes	=	1 K (kilobyte)
1024 K	=	1 M (megabyte)
12 inches	=	1 foot
3 feet	=	1 yard
1760 yards	=	1 mile

Figure 2.10: Conversion

$20M \times \frac{1024K}{1M} \times \frac{1024bytes}{1K} \times \frac{8bits}{1byte} = 167,772,160bits$

Now I should briefly say where the number 1024 comes from. Given that we have ten fingers and ten toes we commonly work with a number system that is base 10 (decimal.) We have the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. If we need to represent a number larger than 9 we have a problem. We do not have a symbol for ten. To solve this problem we need to assign a value to each position in the number. The decimal system has the ones place, then the tens place, then the hundreds place, then the thousands place, etc. We can then reuse the symbols 0-9. For example:

The decimal number 809 is 8 hundreds and 0 tens and 9 ones, or
$8 \times 10^2 + 0 \times 10^1 + 9 \times 10^0 = 800 + 0 + 9 = 809$

Each position is ten times greater than the position immediately to its right, and ten times less than the position immediately to its left.

Computers only have 0 and 1 so they work with a number system that is base 2 (binary.) In a base 2 system there is no symbol for two. Again we need to assign a value to each position in the number. Just as the number 10 is very important in our number system, the number 2 is very important in the computer's number system. The binary system has the ones place, then the twos place, then the fours place, then the eighths place, then the sixteenths place, etc. For example:

The binary number 1011 can be thought of as:
$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 0 + 2 + 1 = 11$

Each position is twice as great as the position immediately to its right, and twice as small as the position immediately to its left.

Figure 2.11 shows the different places for the decimal and binary number systems:

This explains why humans commonly use numbers like 10, 100, and 1000, while computers commonly use numbers like 64, 512, and 1024. 1024 is 2^{10} or 2 multiplied by itself 10 times ($2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$.) Now the numbers in Figure 2.11 got rather large. Figure 2.12 shows the numbers 1-12 represented in both decimal and binary.

Now it is not necessary to remember these numbers beyond the ability to convert between bits and bytes and K and M, but it is important to realize that computers do not work with the same number system that we do. Computers do not do math the same way that we do. In the past, the user of a computer needed to know these

decimal			binary		
10^0	=	1	2^0	=	1
10^1	=	10	2^1	=	2
10^2	=	100	2^2	=	4
10^3	=	1000	2^3	=	8
10^4	=	10000	2^4	=	16
10^5	=	100000	2^5	=	32
10^6	=	1000000	2^6	=	64
10^7	=	10000000	2^7	=	128
10^8	=	100000000	2^8	=	256
10^9	=	1000000000	2^9	=	512
10^{10}	=	10000000000	2^{10}	=	1024

Figure 2.11: Place Values for Decimal and Binary

decimal			binary	
01 =	$0 \times 10^1 + 1 \times 10^0$	=	0001 =	$0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
02 =	$0 \times 10^1 + 2 \times 10^0$	=	0010 =	$0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
03 =	$0 \times 10^1 + 3 \times 10^0$	=	0011 =	$0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
04 =	$0 \times 10^1 + 4 \times 10^0$	=	0100 =	$0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
05 =	$0 \times 10^1 + 5 \times 10^0$	=	0101 =	$0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
06 =	$0 \times 10^1 + 6 \times 10^0$	=	0110 =	$0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
07 =	$0 \times 10^1 + 7 \times 10^0$	=	0111 =	$0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
08 =	$0 \times 10^1 + 8 \times 10^0$	=	1000 =	$1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
09 =	$0 \times 10^1 + 9 \times 10^0$	=	1001 =	$1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
10 =	$1 \times 10^1 + 0 \times 10^0$	=	1010 =	$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
11 =	$1 \times 10^1 + 1 \times 10^0$	=	1011 =	$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
12 =	$1 \times 10^1 + 2 \times 10^0$	=	1100 =	$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$

Figure 2.12: 1 - 12 in Decimal and Binary

other number systems to operate the computer. Today you can use normal decimal arithmetic and the computer will worry about doing all the work of translating it to binary. If you decide to continue with computer science courses there are two MORE number systems you have to learn - base 8 (octal) and base 16 (hexadecimal.)

2.9 ROM & RAM

Computers have two kinds of memory stored in chips on the motherboard. ROM stands for Read Only Memory. RAM stands for Random Access Memory (though you should remember it as Read And write Memory.) ROM is used by the computer to store things that it doesn't want to forget ...things like what to do when the power switch is turned on. ROM contains a lot of very basic information that the computer needs everyday. As people we have information on tasks like breathing, and pumping blood that we need every day. It would be very bad if we went to Math class and by learning 2+2=4 we wrote over the part of our memory that told us how to pump blood. Similarly, in a computer we don't want the user to be able to accidentally destroy important information. The ROM is there only for the use of the computer. It can only be read from. The RAM, on the other hand, is there to be used by the **user**. (In computer jargon, the person who uses the computer is known as ...the user') As you type your term paper into the computer, the words are stored in RAM. The word processing program you are using to write the term paper is also temporarily stored in RAM giving the computer instructions. RAM can be modified by the computer. It can be read from or written to. Both types of memory are measured in bytes. Personal computers today have about 128K of ROM and 1M of RAM.

There is one big difference between ROM and RAM, and that is what happens after you turn the power off. The information stored in the ROM chips will still be there. ROM is **involatile**. The information on the RAM chips will disappear. RAM is **volatile**. The electrical power supplied to the computer keeps the information in RAM. When that power is cut off, the computer "forgets" everything that was in the RAM. ROM is like your textbook. Information has been stored in there permanently by the publisher. You can not alter it. RAM is like the blackboard. A blackboard is written on, modified ...and eventually the janitor comes in and washes all the chalk off of it. The blackboard is there for temporary storage while the ideas are being discussed. The textbook is there for permanent storage.

RAM is also referred to as **Primary Storage** or **Main Memory**, since it is

ROM	read only	permanent
RAM (PRIMARY STORAGE)	read/write	temporary
SECONDARY STORAGE	read/write	permanent

Figure 2.13: Types of Memory

the place where the computer does most of its processing. We use our brains as primary storage, to hold things we are working on and thinking about. You can think of turning the computer off in the same way that we go to sleep. Some of the information we had before went to bed is gone ...forgotten. When we wake up we can still remember basic important things, but we occasionally forget about appointments or phone numbers. What do we do about this? We write those things down in a more permanent form on a piece of paper, or in an address book. We use **Secondary Storage**. Computers also make use of secondary storage in the form of disc drives.

So let's expand our classroom analogy a bit. We have the textbook as ROM (assume that you are a good boy or girl who does not write in your textbooks so you can get high resale value.) We have the blackboard as RAM. We have our notebook as secondary storage. While we are sitting in class we can "read" information from the textbook, the notebook, or the blackboard. We can "write" information on the blackboard or in our notebook. Both the textbook and the notebook are permanent forms of storage. After we leave the classroom we can still "read" information from the textbook or the notebook. The blackboard is only used for temporary storage, while we are discussing something. When we leave the class or the board is erased, we can no longer "read" that information. Now your notebook has a big advantage over your textbook in that you can erase your old notes, and write new ones in their place. You can "read" from and "write" to your notebook, where you can only "read" from your textbook. Figure 2.13 shows the same thing in computer science terms.

2.10 Floppy Discs

When we buy a record album it comes on a compact disc. We put this disc into our CD Player to listen to it. When we obtain a piece of software, it usually comes on a **floppy disc**. Floppy discs come in two popular sizes. They are called **5&1/4"** ("Five

and a quarter inch") and **3&1/2"** ("Three and a half inch.") They are appropriately named since the 5&1/4" disc is 5&1/4 inches on each side and the 3&1/2" disc is 3&1/2 inches on each side. 5&1/4" floppy discs were the standard in the early 80's, but the 3&1/2" became the new standard in the late 80's since they are smaller, more durable, and hold MORE information than their 5&1/4" siblings. Figures 2.14 and 2.15 show what they look like.

All personal computers come with at least one built in **Floppy Disc Drive** (or Floppy Drive for short.) Usually this drive is built into the main unit so only a small slot is visible for inserting the disc into. This is an **internal drive**. You can also buy floppy drives that sit outside of the main unit. One of these is called an **external drive**. As with the floppy discs, the drives come in 5&1/4" and 3&1/2". You then go out and buy floppy discs that you can insert into the drive. Computer users often have hundreds of floppy discs laying around in shoeboxes and scattered in drawers with various information stored on them, but only one at a time can be used in the floppy disc drive. Floppy discs allow you to "write" information onto them, and "read" information off of them. You can also write over any information currently on the disc with new information. They are a lot like cassette tapes in this regard, and early personal computers used cassette tapes instead of floppy discs because they were cheaper.

The price of floppy discs has also dropped dramatically over the last decade. In 1981 a box of ten 5&1/4" discs would cost $50. Today a box of ten 5&1/4" discs costs $2.50 (unless you buy them at the bookstore where you can expect to pay $1.50 per disc. Ain't capitalism wonderful) In 1984 a box of ten 3&1/2" discs would cost $50. Today a box of ten 3&1/2" discs costs $4.50 (unless you buy them at the bookstore where you can expect to pay $2.00 per disc.)

Like record albums or cassette tapes, floppy discs must be handled with a certain amount of care. Each floppy disc contains a circular piece of mylar coated with magnetic oxide. With a 5&1/4" disc this magnetic media is encased in a flexible plastic sleeve with openings so the disc drive can read from, and write to the disc. These openings also allow you to touch the magnetic media. Do not touch the magnetic media. 3&1/2" discs are an improvement on this since the plastic sleeve isn't as flexible as the 5&1/4" variety, so the disc is harder to bend. In fact a 3&1/2" "floppy" disc is not floppy at all. The slots for reading and writing are covered by metal covers to keep out the dust and your fingers.

We need to keep discs away from static electricity, magnetic fields, and dust. So don't rub your cat with a disc, don't put it next to the television, and don't use it as a dust-pan. You also shouldn't set it next to the phone if you think it might ring.

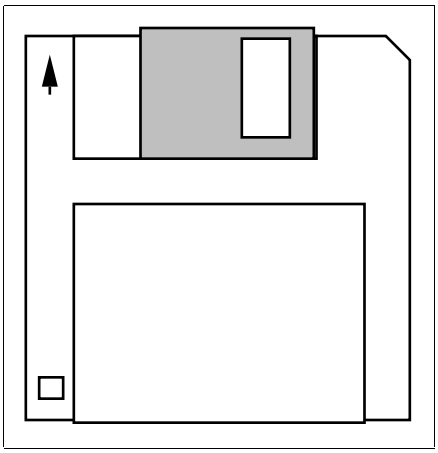


Figure 2.14: 3&1/2" disc (top view)

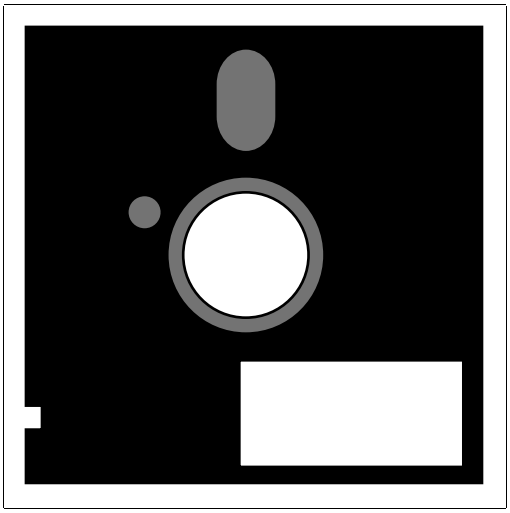


Figure 2.15: 5&1/4" disc (top view)

Don't do anything with a floppy disc that you wouldn't do with a record album or a cassette tape. Don't leave it in the sun, don't bend it, and don't use it as a frisbee on the beach (at least until the class is over.)

Cassette tapes and video tapes have write-protect notches. This is a small piece of plastic that allows you to write on the tape. When the notch is popped out you can no longer write information on to the tape. The tape is "write protected" so you do not accidentally erase the information that is stored on it. Floppy discs have a similar feature. The 5&1/4" disc has a square notch cut in its side. When this notch is visible you can write on the disc. When the notch is covered up you can not write on the disc. The notch is usually covered using a small piece of adhesive tape, so it can be removed at a later time. 3&1/2" discs have a similar, but slightly more convenient approach. They have a slot on the back with a small slider. When this slider is moved so that the write-protect hole is covered, the disc can be written on. When the slider is moved so that the hole is open, the disc can not be written on.

2.11 Hard Drives

One floppy disc can store approximately one Megabyte of information depending on the drive that it is used in. This may seem like a lot of space, and it was only a few years ago ... but not now. Now most personal computers come with Hard Drives (or Hard Disc Drives.) Like floppy drives, hard drives can either be internal or external. Hard drives for computers commonly hold 40 to 100 Megabytes of information. Hard drives are about the same size as floppy disc drives, so how come they hold so much more information? The aluminum platters that hold the information are not removable from a hard drive. The platters are held within a sealed environment so no dust can get onto them. This allows the Hard Drive to spin the disc much faster (floppy drives spin at 300 RPM, hard drives spin at 3000 RPM, where as old 33&1/3 LPs spun at 33&1/3 RPM.) The information can be packed much more densely on the solid platters so they can store more information. Floppy discs are carried around in your pocket, bumped and scraped during normal use, and rubbed against cats by gullible first year students. Hard discs are kept safe within their drive. This means that Hard discs have faster access, longer lifetimes, and lower failure rates than floppy discs.

You may be wondering why are floppy discs called floppy discs and hard discs called hard discs. Floppy discs have a piece of mylar inside. Hard discs have an aluminum platter inside. The mylar is floppy. The aluminum is hard.

Both floppy discs and hard discs have their advantages. Floppy discs don't hold as much information, and are not as fast, but they are portable, and are cheaper. Hard discs are not as portable, and cost more, but they store a lot of information, and are very fast. As with the cost of floppy discs, the cost of hard drives is decreasing rapidly Ten years ago a 10 megabyte hard drive would have cost $5,000. Today you can get a 40 megabyte hard drive for $400. Most serious personal computer users use both a floppy drive and a hard drive. This allows them to have the advantages of both kinds of drives, with the only disadvantage being felt in the pocketbook.

The future of disc drives for computers lies with technology closer to that of a compact disc. Compact discs, and their larger siblings Laser discs can store a phenomenal amount of information. One compact disc can hold a gigabyte of information. One gigabyte is 1024 megabytes. That's a lot. Most personal computers today have a **CD ROM** drive as an optional accessory. This drive reads programs (and music) stored on compact discs. They are called CD ROM because, like musical compact discs, the ability to write on them is reserved for companies. This should soon change. 3&1/2" optical discs that you can read from and write to are already on the market and can store 128 Megabytes on a single disc. The big disadvantage of this system right now is that the Disc Drive costs $3000, and each individual disc costs $130. But as we have seen, prices have a tendency to drop very quickly.

2.12 Modem

Personal computers do not come with a built in modem, but it is one of the more useful pieces of hardware you can add to a computer system. A modem is a piece of hardware that allows your computer to "talk" to other computers. The modem is used to convert the information in the computer into sounds that can be sent over the phone line. The person on the other end of the phone line needs a modem to convert the sound back into information for her computer to understand. Fax machines are currently becoming very popular, and they work on a similar principle. Some modems sit inside the main unit on cards, while others are boxes that sit next to the phone and are attached to the computer with a cable.

Memory and discs are measured in bytes, but modems are measured in terms of their **Baud Rate**. A typical modem in the early to mid 80's was 300 ("three hundred baud"), in the mid to late 80's was 1200 ("twelve hundred baud"), and in the late 80's and early 90's 2400 ("twenty-four hundred baud"). The higher the baud rate the faster the modem can send information across the phone line.

2.13 Back to Bytes

You may think that all this bits and bytes and K and M talk is just here so we can ask exam questions. Its not. If you open up the Detroit News or Free Press you will see ads for computers, and they will all say how many K of this or how many Megs of that their computers have. Obviously each company is going to try to make their computer look better by picking the best unit of measure. Its good to be able to compare these machines on common ground. The machine with 512K of main memory may sound better than the one with 2M of main memory. After all, 512 is a much bigger number than 2, but the 2M computer has four times as much memory as the 512K computer.

Today, the number of colours a computer can display on its monitor is usually described in terms of bits. Some computers have "8 bit colour" which means they can display 2^8 colours simultaneously. That is $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 256$ colours on the screen at one time. More expensive personal computers can display "24 bit colour" which means they can display 2^{24} colours simultaneously. That is $2 \times 2 = 16{,}777{,}216$ colours on the screen at one time. Do you really need 16 million colours? Probably not, but they sure look pretty.

2.14 Variety

So far we have talked only about personal computers. Computers range in power, size, and price from **MicroComputers** to **MiniComputers** to **Mainframes** and up to **SuperComputers**. Personal computers are usually Microcomputers but some are getting near the power of MiniComputers. Personal computers are designed to be friendly. Mainframes and Super computers are designed for raw computing power. While you can sit down in front of a Microcomputer or a minicomputer and bang on the keyboard, you can only talk to Mainframes and Supercomputers through other smaller computers. Complex calculations and sophisticated graphics require the use of a lot more computing power than a personal computer can offer. A personal computer could do the same job, but it would take much longer to do it. Figure 2.16 compares these different types of computers.

Criteria	Micro	Mini	Mainframe	Super
Price	$2,500	$100,000	$1 million	$10 million
Users	1	10	500	?
Power	1	×10	×100	×1000
Made by	Apple IBM Commodore	Apollo Sun DEC	IBM	Cray
Used By	anyone	CS students Ford	Wayne State IRS Comerica	Pentagon NASA
Used For	Writing Budgeting	Research CAD/CAM	Databases big calculations	weather prediction large simulations

Figure 2.16: Comparison of Computer Types

2.15 Brief History

The first big computers were built in the early 30's, but were mostly curiosities until World War II. The US Army wanted a device to perform trajectory calculations. In 1946 the ENIAC was completed. ENIAC stood for 'Electronic Numerical Integrator And Calculator.' It weighed 30 tons and occupied 1500 square feet. It was programmed by moving wires and throwing switches. John vonNeumann improved on this design with the EDSAC in 1949. EDSAC stood for Electronic Delay Storage Automatic Calculator and was the first computer to use a program stored in memory. Since then the government, universities, and big businesses have used computers and big companies such as IBM have supplied them. The first experience most people had with computers in their home came when Atari began making the "Pong" game in 1971. None of the companies making large computers at the time thought there was any market for a personal computer. Personal computing was a hobby. You bought the kit and tinkered the machine together and then you tried to figure out some use for the thing. The first machine to actually be called a "personal computer" was the MITS (Model Instrumentation Telemetry Systems) Altair 8800. It was released in January 1975 and it sold for $400. It had a whopping 256 BYTES of memory, no

monitor, no keyboard, no secondary storage. The machine was sold by mail-order and you could buy the kit like most other "micro computers" of the time, or, and here was the new concept, you could buy it pre-assembled. It did cause a bit of a stir among hobbyists.

The industry really didn't get started until 1977 (the year "Star Wars" was released,) with the introduction of the Apple][, and a few months later the TRS-80 Model I (TRS standing for Tandy Radio Shack) and the Commodore PET (PET standing for Personal Electronic Transactor.) These computers had the advantage of being sold in stores. Imagine that - a personal computer sold in stores. IBM entered the market four years later and almost immediately became the de-facto standard (the "big blue" light blinds many businessmen.) With the introduction of the IBM-PC, businesses began to take personal computers seriously, and by 1983 IBM practically owned the market. In 1984 the Macintosh arrived and computers became 'intuitive' to computer illiterates. In 1985 the Amiga came on the scene with enhanced graphics, sound and the ability to run more than one program at the same time. Since then, improvements in personal computers have come at a slow and steady pace. In 1990 there were 9 million personal computers sold in the United States. 18/Apple, 6/

The last 15 years have seen a huge number of personal computers being released. Some would rapidly fade into obscurity (the Apple III, the Osborne, the Adam, and the PCJr), and some would hit the big time (the Apple II, the TRS-80, the Atari 800, the VIC-20, the IBM-PC, the Macintosh, and the Amiga). Figure 2.17 lists many of the personal computers that have been released over the last fifteen with the more important machines are shown in bold-face. Figure 2.17 also lists some popular theatrical released in the same years to give you some bearings.

2.16 Comparison

To give you some idea of how fast computers are improving we will compare a few different models. Figure 2.18 shows the features of several "low end" (i.e. under $2000) personal computers which were released at approximately three year intervals. Computers are one of the few areas where you get more features year after year and the price continues coming DOWN as the machines become more popular. Figure 2.18 shows the features at the time of initial release, although all of these machines have gone through various upgrades.

Year	Movie	Apple	Commodore	IBM	Tandy
76	Robin & Marian	Apple I			
77	Star Wars	**Apple][**	PET		**Model I**
78	Animal House				
79	Alien	][plus			Model II
80	Blues Brothers	Apple III			Model III
					Color Comp.
81	On Golden Pond		**VIC-20**	**IBM-PC**	
82	Poltergeist		C-64		
83	Right Stuff	Lisa		PC XT	Model IV
		][e		PCjr.	Tandy 2000
84	Buckaroo Banzai	**Macintosh**		**PC AT**	Tandy 1000
		][c			Tandy 1200
85	Fright Night		C-128		
			Amiga 1000		
86	Top Gun	Mac Plus			Tandy 3000
		][gs			
87	Princess Bride	Mac II	Amiga 500	PS/2	Tandy 4000
		Mac se	Amiga 2000		
88	Die Hard	Mac IIcx	Amiga 2500		
		Mac se/30			
89	Glory	Mac IIci			
90	Ghost	Mac IIfx	Amiga 3000	PS/1	
		Mac IIsi			
91		Mac lc			

Figure 2.17: 15 Years of Personal Computers

	Release Date	Clock Speed	RAM	ROM	Secondary Storage	Price
Altair 8800	Jan '75	?	1/4K	0K	none	$400
Apple][	Apr '77	1 MHz	16K	8K	Cassette	$1,200
IBM-PC	Aug '81	5 MHz	64K	40K	5&1/4" 160K	$2,500
Macintosh	Jan '84	8 MHz	128K	64K	3&1/2" 400K	$1,800
Amiga 500	Feb '87	8 MHz	512K	128K	3&1/2" 880K	$1,400
IBM PS/1	Jul '90	16 MHz	1M	256K	3&1/2" 1.44M	$1,600

Figure 2.18: Improvements over 15 Years

2.17 Clones

You may have heard of clones. While the ability to clone human beings is still a "few" years off, clones of computers have been among us for almost a decade. There have been Apple][clones and IBM-PC clones. A clone is a computer built by company X that runs just like a machine built by company Y. Usually the clones run even better than the original, and cost less as well. Is company Y upset about this? Damn right they are, and out comes their legal department. Unfortunately for company Y, most clones are perfectly legal. The IBM-PC has many, many, many clones out there, and they sell extremely well.

Most computer companies do not build their own hardware, but buy it from other companies that specialize in a certain product (such as a CPU or a monitor, or a disc drive.) The computer company puts all these parts together, writes the operating system, and then announces to the world what they have created. A company that manufactures clones can buy the same parts from the same companies, write their own operating system (throwing in a few new features,) and they have a ready made market for their machine.

Clone manufacturers can charge less because they do not have to support the research and development costs to actually design the machine. They know that the machine works because it has already been built by someone else. Now all they have to do is build their own.

Now there are several drawbacks to buying a clone. The first is that you do not get the support that a big company can offer. Companies like IBM, Apple, Tandy, Compaq, and Commodore have large networks of dealers to service their products.

The clones do not have this kind of support. The companies that build clones are also much more likely to disappear than the major personal computer manufactures. Upgrades are harder to come by for clones as well. Clones can be cheaper in the short run, but possibly more expensive in the long run.

2.18 Hardware and You

So what have we learned so far. We now know that mice do not have legs, but chips do; that floppy discs are not necessarily floppy; and that personal computers are sold in stores. We also suspect that computer scientists spend most of their time thinking up acronyms so no one will know what they're talking about. Who says computer scientists don't live in the real world?

Around this time, computer science students begin to ask me what kind of computer I think they should buy. This is actually a very difficult question because it goes far beyond tables of numbers and lists of features. The following are a list of questions in descending order of most importance in choosing a personal computer.

1. Can I afford it? - Don't let anyone kid you. Price is the most important thing to keep in mind when buying anything.

2. Will I be compatible? - If the place where you work or go to school uses mostly one type of computer, and you want to work at home, that is the brand you will probably have to buy.

3. Can it do what I need it to do? - You are buying a personal computer to help you in some way. Be sure that it has the hardware and the software (coming next chapter) necessary to do what you need it to do. Are you buying the machine to do word processing, or graphics, or to run business applications? Each personal computer has strengths in different areas.

4. Is it easy to use? - You want to spend time doing your work not figuring out how to do your work

5. Do I have a friend with this kind of computer? - You usually end up buying a computer because you know someone else who has one. Asking your friends is the best way to get good advice about a particular machine before and after you buy one. Friends are also good sources for "borrowing programs to try them out."

6. Is it expandable? - This will be more important in the long-term, than the short but you should look for machines with expansion slots, and other ways you can upgrade in the future. This way you can buy only what you need now, and expand later.

You should try a computer out before you buy it, preferably not with a salesperson lurking over your shoulder. This is where having friends with computers really helps because you can go over and bang on their machine for a while. The important thing is to keep an open mind ... look around ... see for yourself. We all have different tastes in movies and music. Computers are no different. You should buy the computer that will help YOU the most. Keep in mind that this may not turn out to be a computer at all. Maybe all you need is a typewriter, or a address book, or a big pad of paper and a handful of coloured markers.

2.19 Questions

1. What is the difference between RAM and ROM? What is each used for?
2. Why is it necessary to have part of memory be 'read only'?
3. How can a dot matrix printer print out both text and graphics?
4. How many bits are in 40M?
5. What do we measure in bytes?
   (a) the speed of the computer
   (b) the amount of memory in the computer
   (c) the weight of the computer
   (d) the number of CPUs in the computer
6. What is secondary storage?
   (a) RAM
   (b) ROM
   (c) where you put your computer when you are done using it
   (d) floppy discs

7. What does the RGB in RGB monitor stand for?

   (a) Randomly Generated Bandwidth
   (b) Raymond Gerald Brown (the inventor)
   (c) Red Green Blue
   (d) Rotating Gauss Band

8. A 3&1/2" disc is called a 3&1/2" disc because

   (a) it holds 3.5 times as much information as a 5&1/4" disc
   (b) its length and width is 3&1/2 inches
   (c) you can fit 3&1/2 of them into a single drive at the same time
   (d) it spins 3.5 times faster than a 5&1/4" disc

9. Which of the following is volatile?

   (a) RAM
   (b) ROM
   (c) Floppy Disc
   (d) Hard Disc

10. Why are floppy drives better than hard drives?

    (a) floppy discs are made of aluminum
    (b) floppy discs can be removed from the drive
    (c) floppy discs can store more information
    (d) floppy discs are faster

11. What does a MODEM allow your computer to do?

    (a) access hard drives larger than 1024M
    (b) display colour graphics on your monitor
    (c) print out graphics
    (d) talk to other computers over telephone lines

Chapter 3

Software

Software is information. It is words and numbers. Like music on a cassette tape, it is invisible. So far we have been a little vague on how the software connects to the hardware. The computer's operating system is a piece of software. This set of instructions is stored in the ROM chips. When the computer is turned on, the computer knows to look at the ROM chips for instructions as to what to do first. It then follows these instructions. The computer is now ready for you to give it some specific job to do. To give it these new instructions, we need to supply it with more software.

3.1 Programs

When you get a piece of software, it will usually come on a floppy disc. You put this floppy disc into your disc drive. The computer's operating system will allow the computer to read the program on the floppy disc and put the copy of it into the RAM. This program could be a word processor. It could be a spreadsheet. It could be "Space Invaders." Whatever the program is, it needs to be in RAM for the computer to execute the instructions contained in it. Now we have the ROM controlling the basic, "lower level" functions - and the RAM containing a program to do some "higher level" functions - like writing a term paper. The program (in RAM) talks to the operating system (in ROM) when it needs to, so you do not have to. Figure 3.1 illustrates.

Why are things set up this way? The answer is generality. We could have one computer specializing in word processing, another specializing in spreadsheets, and

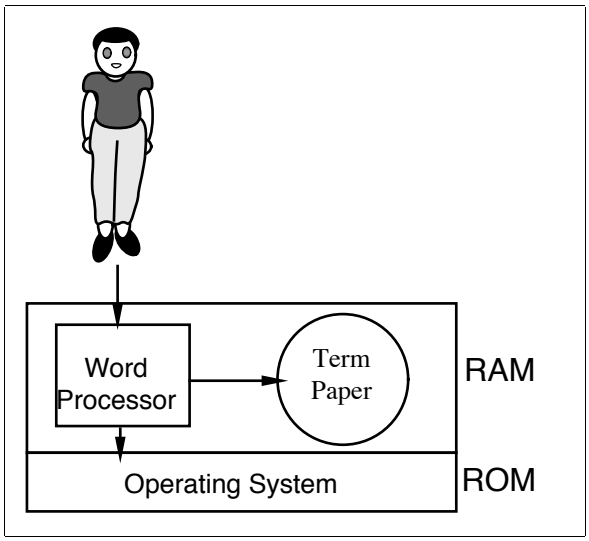


Figure 3.1: Software

another specializing in database where all the necessary information is stored in the ROM. Machines such as these do exist, but it would be expensive to buy all three. What we want is one all purpose computer that can do all of the above and more. Think about your stereo system. When you buy your components they do not come with built in music. You don't have a rock stereo, and a classical stereo, and a rap stereo. You have one stereo that can be used to play all different kinds of music. This gives you the added advantage of being able to play new music just by buying the CD and popping it into your player. When we want to run a new piece of software we buy the floppy disc, pop it into our disc drive, and run the program.

The computer's ROM contains instructions that all programs need to run . . . instructions like how to run the disc drive, monitor and keyboard. The RAM is used to store the specific program that is being used. We can put a word processor into the RAM, or a database, or a spreadsheet or "Space Invaders." You need to open this Course-Pak and read the words into your brain before you can think about them. Once you have read the words in you can process what you have read. The computer must read the program off the disc and store it in it's RAM so it can process it.

The size of software programs has grown dramatically in the last decade. Ten years ago a good word processor might take up 100K. Today a good word processor takes up 700K. Since we need to be able to load the program into memory it is important that we have enough RAM to store it. But when we are writing a term

paper with our word processor, the word processor is not the only thing in the RAM. The term paper is stored in the RAM as well while we are working on it. The RAM holds the data being processed - the term paper, as well as the program doing the processing - the word processor.

More and more features are added to programs each year to attract new buyers. Usually a software company will continue to offer new versions of their program for several years. People who bought the early older version are usually given the opportunity to upgrade their program (for a small fee, of course) to give them access to all of the new features. Software companies tend to work very closely with the computer manufacturers since the software company wants to be ready when a new model arrives.

Usually a specific piece of software is written for one specific machine, and if the product does well, the program will be rewritten for another machine. For example, Microsoft wrote Excel for the Macintosh. It was a big hit. Then they wrote Excel for the IBM-PC. Conversely Word Perfect was a big hit on the IBM-PC, so a version was written for the Macintosh. While these programs look similar to the user, they are running on computers with different CPUs.

People use personal computers for many different reasons so there is a wide variety of software available. Some programs are business oriented such as Word Processors, Spreadsheets, and Databases. Some are creativity oriented such as Paint Programs, Animation Programs, and Music Programs. Some help you write programs yourself such as Compilers and Interpreters. Some help you communicate with other computers such as Terminal Programs and BBS Programs. Some help you to relax such as videogames.

Figure 3.2 lists some of the more important pieces of software that have been released within the last ten years. These programs were either the first of their kind, or the standard that was followed for similar programs.

3.2 Piracy

Piracy has been a problem for the computer industry since the first programs came out. A pirate is someone who illegally copies a piece of software. Software companies charge hundreds or thousands of dollars for their programs, and there are people who do not feel like paying that much. In fact they do not want to pay anything. It is easy to make an exact copy of a program stored on disc, all you need is a copy program and a blank disc. Of course you would have to go and photocopy the instruction

Year	Word Proc.	Database	Spreadsheet	Graphics	Hypertext
79	Word Star		VisiCalc		
80					
81		DBase II			
82			Lotus 1-2-3		
83					
84	MacWrite			MacPaint	
85			Excel		
86					HyperCard
87					
88					

Figure 3.2: Ten years of Software

manual, but that doesn't slow the pirates down much.

The software companies and the pirates were soon locked into a vicious circle. The companies would add better copy protection to their programs to make the programs harder to copy. They would then charge more for the program to pay for this new protection. The pirates would soon find a way around the new protection. Users felt less inclined to pay the higher price - so more people pirated the software. Eventually the copy protection schemes got to be intolerably annoying to every user, including those who payed for the program. Most users make a backup copy of the discs they buy, or move the programs onto their hard drive. Many copy protection schemes made this impossible. The companies came up with a new strategy: sell the program cheaply without any protection and people will be more inclined to pay for it. This new strategy has worked very well.

3.3 Public Domain Software

One of the reasons that the software companies decided to drop their expensive protection schemes was that public domain programs were taking their business. A public domain program is free for you to use, copy, and give to anyone you know. The program is literally in the public domain. There is a large amount of public domain software out there, and some of it is as good or better than professional

programs. Public domain programs are most often written by "average" people who need a program for a certain task, and then allow others with the same need to use it.

3.4 Emulators

We said before that the computer's operating system handles all the nitty gritty, low-level work while the programs you buy deal with more high- level work. One interesting program you can buy is called an emulator. An emulator is a program that converts your brand of computer into another brand of computer. You do not need any extra hardware, because the software is fooling the programs into thinking that they are running on the machine they are supposed to be running on. In effect you get two computers in one. You can run software for your machine, and software for the machine you are emulating. Software emulators usually run slower than the machine they are emulating, but the emulator software is much cheaper than the corresponding amount of hardware.

Another type of emulator requires hardware from the computer that you are emulating. Commonly the CPU and the ROMs from the machine you wish to emulate are put on a card that you can plug into your computer. These emulators work much faster than the plain software emulators, but you have to pay more to get the necessary hardware.

3.5 Software and You

Software is just as important as hardware. One of the reasons that there is no such thing as a 'best' personal computer is that they all run different software. It is the combination of hardware and software that is important when selecting what computer to use for a specific job.

Chapter 4

Communicating with the Computer

We said before that computers operate in a world of 1 and 0, true and false, yes and no, on and off. Human beings operate in a world of "maybe" and "kind of." The computer is digital. We are analogue. In order to communicate with the computer and tell it what to do we use what is called an **interface**. The interface "sits" between us and the machine as show in Figure 4.1. The keyboard and monitor are the most common interface. Information passes through this interface between us and the machine. In the past this interface has been very close to the machine's way of working, and very far from the way human beings work. Graphical User Interfaces, and alternate input devices were created to move the interface closer to us, to make the machine easier to use.

4.1 I/O

I/O is short for Input/Output. We give input to the computer and it responds by giving us some output. This is pretty straight-forward since input goes INto the computer and output comes OUT of the computer. The keyboard is the most common input device, and the monitor is the most common output device. Modems and Disc Drives are used for both input and output. Figure 4.2 lists several others.

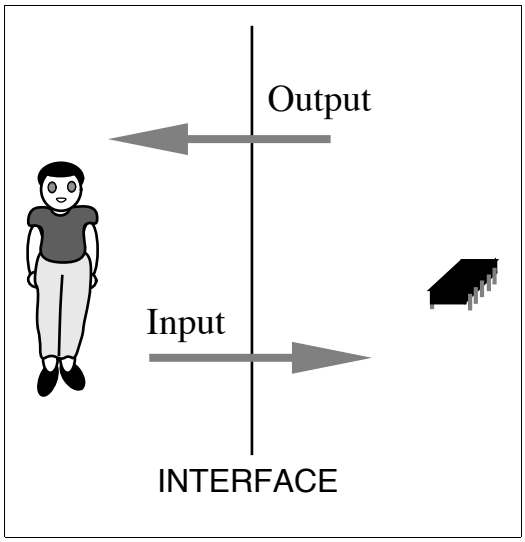


Figure 4.1: Interface

Hardware	Input	Output
Keyboard	yes	
Mouse	yes	
Joystick	yes	
Monitor		yes
Speaker		yes
Printer		yes
Modem	yes	yes
Disc Drive	yes	yes
Tape Drive	yes	yes

Figure 4.2: Input and Output

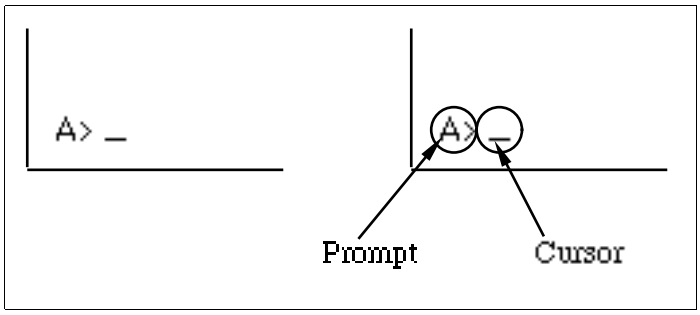


Figure 4.3: Command Line Interface

4.2 Command Line Interface

The personal computers of the 70's and early 80's used the keyboard as the main way for the user to tell the computer what he wanted to do next. The computer displayed a blank screen with a small prompt and a flashing cursor. The prompt prompts you for a command and the cursor sits there blinking, telling you that the computer is waiting for you to type something in, as shown in Figure 4.3.

The user types in some command and the computer then goes off and executes it. There is a big problem with this method - the user must either memorize the command or constantly be looking through the manuals that came with the machine. This saved the computer a lot of work since the user was forced to be very exact and very specific. Computers are very fast, but they are not very "smart." They must be told exactly what to do. Typing commands at a prompt is called a Command Line Interface (CLI), and remains very popular today with operating systems such as **UNIX** and **MS-DOS**. Once you have learned all the commands you can work very quickly, but it is very frustrating for beginners.

4.3 Input Devices

The standard alphanumeric keys on the keyboard are just one way of telling the computer what to do. They are very good for typing in papers. They are not very good for drawing artwork or playing games. There was soon a need for devices beyond the standard keyboard. Some of these devices included **Arrow Keys**, the **Joystick**, the **Light Pen**, the **Graphics Tablet**, the **Touch Sensitive Screen**, the **Mouse**, and the **Trackball**. Each of these devices has its own advantages and disadvantages

depending on what job they are asked to perform. Currently most personal computers on the market make use of arrow keys, joysticks, and mice as well as the standard keyboard. Eventually we will be able to add speech and writing recognition to this list, but those are some years away.

Videogames are generally best played with a joystick or a trackball. These devices allow rapid, if imprecise, movement.

Drawing artwork is generally best done using a graphics tablet. A graphics tablet is a board that sits on your desk, and drawing on the board causes a similar event to appear on the computer screen. For example, if you pick up the stylus for the graphics tablet and draw a line, a similar line will appear on the computer screen. Graphics tablets allow very fine control since we are physically drawing using a pencil-like object.

Selecting objects on the screen is best done with a light pen, a touch sensitive screen or a mouse. A light pen is a pen-shaped object that is held up to the screen. The light pen reads signals sent out by the monitor and tells the main unit, based on the signal it is receiving, what option is being selected. Light pens do not work very well in bright rooms, and need to be held at specific angles to the screen.

A touch sensitive screen looks just like a regular monitor, except that it is sensitive to pressure. If you press on the screen, the screen can locate the point you pressed and relay this information back to the main unit. If you saw "Die Hard," you might remember that the Nakatomi Plaza complex had a touch sensitive screen in the lobby that Bruce Willis used to locate his wife. Touch sensitive screens have the problem that you really don't want people touching the monitor. Fingers tend to be greasy so the screen must be cleaned more often. Someone may press a little too hard. Fingers are also fairly bulky, so they are not good for very fine work on the screen.

One of the most recent attempts at a friendlier interface is a combination of the light pen and the touch sensitive screen. These new pen interfaces use handwriting recognition and gestures with the pen to interact with the computer.

The mouse is currently the best compromise device for doing general work. Along with the mouse on your desk there is a small pointer on the monitor screen. This pointer moves on the screen as you move the mouse on your desk. You move the mouse to the left, the pointer moves to the left. This pointer acts like your surrogate hand on the desktop. Pressing the button(s) on the mouse is like using your hand to select, or grasp something on the monitor screen.

Say you wanted to draw a circle on the screen. With only a command line interface this would probably involve typing a cryptic command such as DRCIRC(214, 56, 7) giving the center points coordinates and the radius we desire. Well, that's not really

very intuitive. This is not the way that we draw circles. It makes things very easy on the computer, but not easy on us.

Some improvement can be obtained using arrow keys. Arrow keys would allow us to draw a circle made up of horizontal and vertical lines on the screen. This is much closer to what we think of when someone asks us to draw a circle. Even though we are still typing keys on the keyboard. It is much more analogue.

The mouse, or a graphics tablet makes the job even more intuitive. With these tools we actually draw a circle - we move the mouse on the table or the stylus on the graphics tablet in a circular motion as if we are drawing with a real pencil. There is no longer a difference between drawing on the computer screen and drawing on a piece of paper. The computer is now "letting us" do things the way that is easy for us.

Very slowly the interface has been moving away from the computer and closer to the user. The computer becomes easier to use, because now the computer expects us to do things the human way. Learning how to use a computer is easier now because we do not have to learn very many new things. As with hardware and software there is no 'best' interface. The 'best' interface depends on the job being done and the person doing it.

4.4 Graphical User Interface

The most noticeable improvement in user interfaces for personal computers came in 1984 when Apple introduced the first Macintosh. The Macintosh was the first successful personal computer with a mouse, and a Graphical User Interface instead of the then standard Command Line Interface. The Graphical User Interface (GUI for short) concepts were developed in the 70's at Xerox's Palo Alto Research Center. Xerox released the Star Information System with a GUI in 1981 and charged $30,000 for it. It didn't sell. Apple released the Lisa in 1983 and charged $10,000 for it. It didn't sell. Finally, in 1984, Apple released the Macintosh at $2000. It sold. It sold very well. More importantly, it influenced other personal computers. Today, all personal computers offer a graphical user interface as an option. This type of interface has many advantages, all designed to make the act of computing easier and more obvious.

Instead of a prompt and a cursor, a GUI gives you a small pointer which mimics the motions you make with the mouse. This is used to make selections from the menus or move "objects" around the screen as shown in Figure 4.4. These GUI systems are

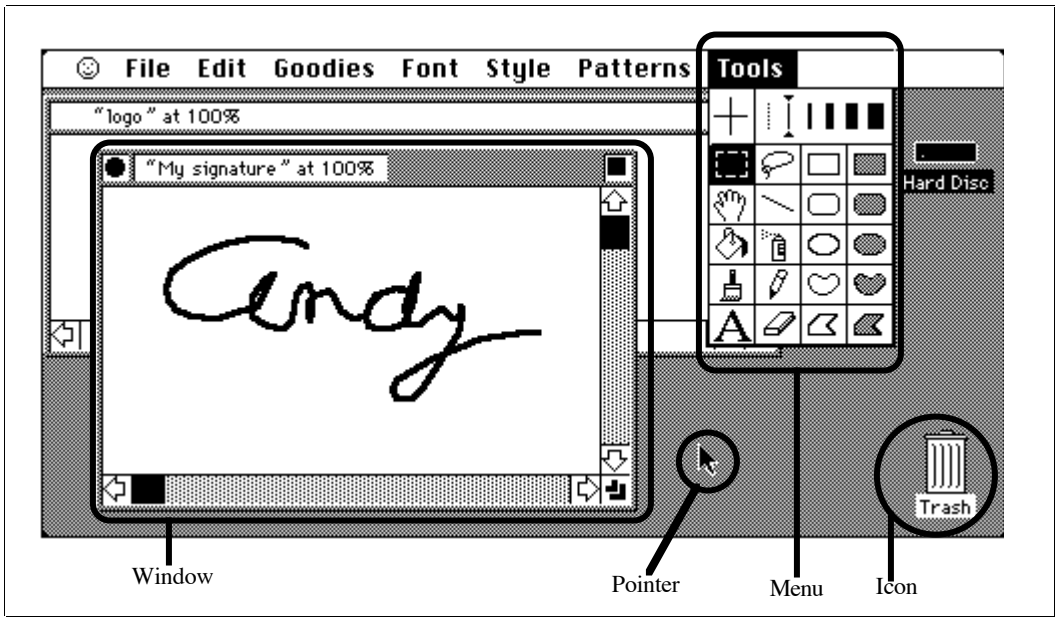


Figure 4.4: Graphical User Interface

occasionally called WIMPs (Windows, Icons, Menus, and a Pointer.) This term was probably coined by someone who thinks 'real' computer users only use command line interfaces.

What are some of the advantages of a GUI?

- Pictures (called icons) replace words for representing ideas.
- Different programs have similar controls.
- Options are presented, and the user chooses from those options
- The user is less aware of the computer's existence.

Companies have used icons for many years, except they call them logos. These pictures represent a product or a company. The benefit of pictures for representing ideas has been shown for years on America's roads. A stop sign has a distinctive shape and colour, as does a yield sign. We do not need to read the words on the sign, because they are not necessary. The picture says it all. Similarly for a traffic signal. We see the colours red, yellow, and green. These coloured lights have a meaning to us, yet no words are there. Imagine if all the road signs were white square signs with

black letters ... some say "stop", some say "yield", and some say "Danger! Blasting Ahead!". We would have to actually read all the signs. People find icons easy to work with on a computer. GUIs have icons. CLIs do not.

There is consistency among stop signs and traffic lights. When you go to another state, stop signs still look the same; a green light still means go. This makes the "rules of the road" much easier to learn. Imagine if we had to learn a whole new set of rules when we cross a state line. That is why having a consistent set of controls in different computer programs makes learning to use the programs easier. GUIs have much more consistent controls than CLIs.

Along with the consistent controls came consistent formats for storing data. This allowed the user to move information between programs, such as copying a picture she drew in a paint program and pasting it into a paper written with a word processor. GUIs make moving information between programs much easier.

When we go into a restaurant we are given a menu listing what is available. We then make selections from that menu. Imagine entering a restaurant for the first time and being forced to blindly guess what they are offering. We might get very hungry before we actually found an item that was available. Allowing the user to choose from options makes learning how to use a computer easier. GUIs have much better menus than CLIs. The menus of a GUI are more consistent than those of a CLI as well.

The user usually doesn't care how the computer does its job just as long as he can get his job done using it. The user wants to feel comfortable interacting with the machine. The user wants to feel like he is on familiar ground, doing familiar things. Most GUIs make use of a desktop. This desktop is designed to look like a real desktop with icons representing physical objects on the real desktop. Icons representing discs and documents have similar looking counterparts in the real world. To throw away a piece of paper in the real world you pick it up, and throw it in the trash can. To throw a document away with a GUI you often move your surrogate hand over the document on the desktop, click the mouse button to grab it, drag over to the trash can, and release the mouse button to let go of it. The actions are familiar. The user doesn't care that the computer is deleting the file from the directory on the disc, he just knows that he has thrown away that document. The user is less aware of the computers existence allowing him to concentrate on the work he wants to do, rather than the fact that he is doing the work on a computer.

Now you may have noticed that many of the advantages of a GUI have little to do with graphics, but graphical user interfaces are much more than icons and menus. This is why they have become so popular recently. Figures 4.5, 4.6, 4.7 show several different graphical user interfaces in use today.

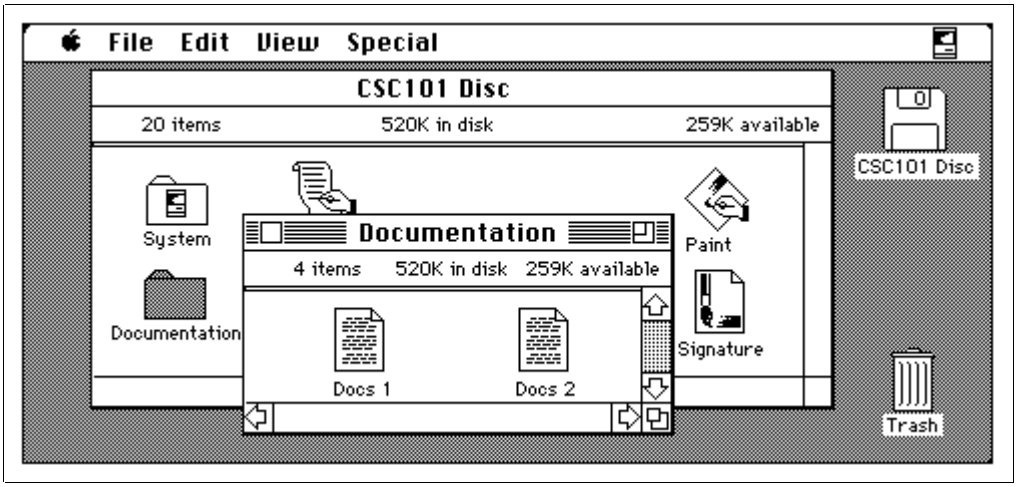


Figure 4.5: The Mac's GUI

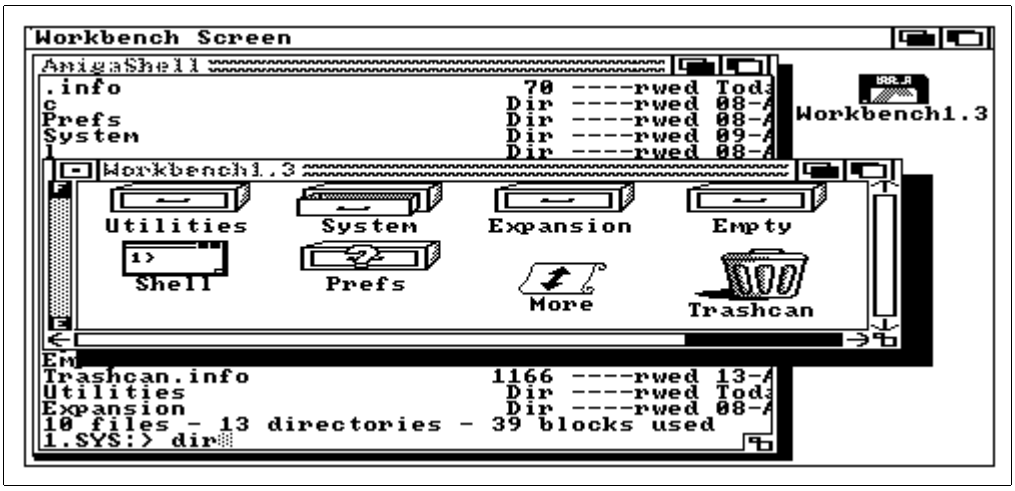


Figure 4.6: The Amiga's GUI

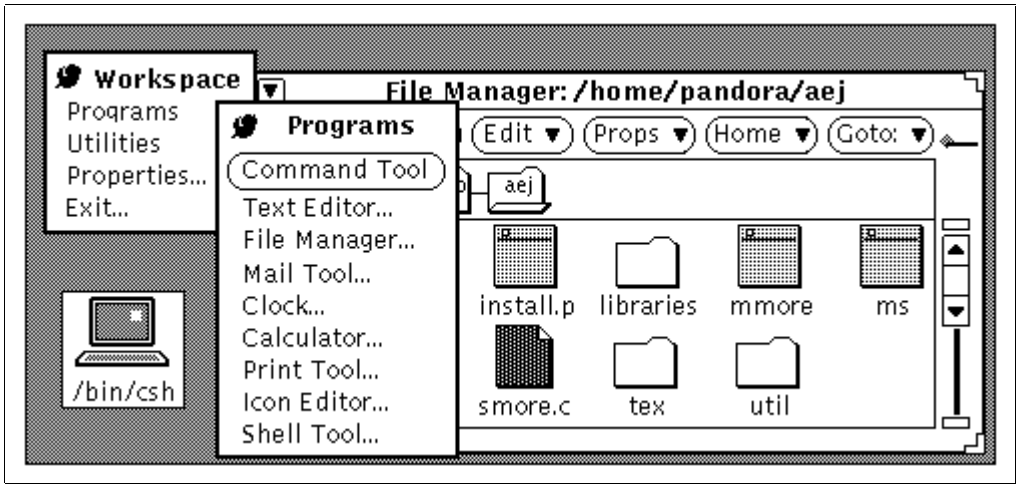


Figure 4.7: The Sparcstation's GUI

4.5 Comparison

There are some people out there who like command line interfaces. There are some people out there who like bell-bottoms, disco, wide collars, and other dated relics of the 70's as well. I am a great believer in graphical user interfaces, but I must admit that there are times when a GUI is a little too helpful and makes my job more time consuming. A graphical user interface acts like an automatic transmission on a car. It allows you to forget about the details of switching gears, and makes a car easier to drive. You may become an expert at driving and decide you would rather have a stick-shift and do the shifting manually. You can do things a little faster, and probably increase your performance.

Each person has the choice of which kind of operating system they prefer. Most students find the graphical user interface much easier, and faster to use. If at a later date, you become more comfortable with computers, you may want to switch over to a command line interface.

4.6 Multitasking

Another improvement in interacting with the computer is a feature called multitasking. Multitasking allows the computer to run several programs (or tasks) at the same time. As human beings, we tend to be working on more than one thing at a time.

We may be writing a term paper and doing some calculations for a lab report. With a multitasking computer we can have the computer calculating the results for our lab report in the background at the same time we type in our term paper in the foreground. This allows for a more efficient use of the machine.

Now "officially" you can not multi-task without having more than one CPU, where each CPU runs a different task. The term has come to include computers which give each task a share of the CPU in turn. The programs are not actually running at the same time, they are taking turns using the CPU. But these turns each last a fraction of a second, so from the users point of view the programs are running at the same time. Since the CPU is being shared, each task will run slightly slower than it would if it were alone. The more tasks you ask the computer to do at once, the slower they will be done, since each will get less time with the CPU.

There are two different types of this 'fake multitasking' in use on personal computers today. The Amiga uses **Preemptive Multitasking** where the operating system assigns how much priority each process will receive. The Macintosh uses **Cooperative Multitasking** where the foreground process decides how much time it will give to the background processes.

4.7 Ergonomics

Of course the purpose of all of this is to make you want to use the computer. This brings up the topic of ergonomics, or human engineering. Human beings design things to be easy for human beings to use. Think about door-knobs, and telephones, automobiles, and a mouse. These devices are designed to be easily used by the average person. The computer manufacturers can not deal with everything however, and there are some simple things you can do to improve your relationship with your computer.

Eye strain and muscle strain can both cause problems. All monitors have controls so that you can adjust their brightness. The "correct" brightness depends on the individual user, and how bright her room is. You should also adjust the monitor so that you don't have a bright light either in front of you or behind you. This causes glare which in turn causes headaches. Another problem comes from sitting, and staring at the screen too long. At least once an hour you should stand up, and walk around a little. This will help your muscles and your eyes. It can also help relax you. These suggestions should sound familiar if you have ever driven a car for long distances at a time - which brings up another important suggestion. If you get tired - get off the computer, rest, have something to eat, and come back later. This way

you won't hurt yourself or your project.

4.8 The Interface and You

Using a computer should not be a painful experience, either mentally or physically. Learning how to use the machine may challenge your mind and your muscles for a short time but there is no substitute for experience, and experience only comes with practice. Newer and more intuitive interfaces should continue to decrease the time it takes for you to become proficient with a new machine, or a new piece of software.

The interface is where we can expect to see the biggest improvements in the near future. Three-D displays are already being demonstrated. Primitive handwriting and voice recognition systems are in use today. While the hardware is constantly being upgraded, most of these improvements are invisible to the user. Improvements to the user interface are immediately apparent to the user, and the easier a machine is to control - the more likely a novice will want to use it.

A new buzzword is 'virtual reality' where the computer generates and manages a nonexistent world that we can interact with. In most videogames, the player has a persona on the screen that we move around and control. We are moving this character through a virtual world, but its not very impressive because the only contact we have with that world is through the monitor screen and possibly some speakers. Enhancements such as the 'power glove' bring us further into that imaginary world, allowing us to interact at a much finer level. The ultimate realization of virtual reality would be something like the holodeck on "Star Trek." Of course that kind of system is a ways off, but important steps are being made in that direction. Computers are become more interactive every day.

4.9 Questions

1. What are two differences between a command line interface and a graphical user interface?

2. What are two similarities between a command line interface and a graphical user interface?

3. What are the benefits of being able to move information between different programs? Can you give an example of where this would be helpful?

4. What is the pointer used for?

Chapter 5

DOS

Given that we have some information stored on the disc, it would be helpful if the computer could access it for us. We need the computer to be able to read from the disc and write to the disc. The physical part of this reading and writing is done by the disc drive, but the CPU must give the disk drive instructions as to what to store and where on the disc to store it. We said that a computer has an operating system. Well, every computer has another operating system to manage information stored on discs. This is called the DOS (rhymes with boss), which is short for Disc Operating System.

5.1 Formatting

While the types of discs used by different companies are the same, each brand of computer stores data on disc in a unique way. Imagine if each VCR manufacturer had their own speed. One company uses SP, one uses LP, and a third uses SLP(EP), but none of the machines can read the other's format. You can use the same blank videotape in each of the machines, but once you have recorded on it you can not play it back on one of the other machines. This is true for floppy discs and hard discs. Each brand of computer "Formats" its discs in a special way. Formatting a disc sets it up to receive information. Information on discs is stored in concentric rings called tracks. Each track is divided up into several sectors as shown in Figure 5.1. Formatting a disc sets up these tracks and sectors. Once a disc has been formatted for a certain brand of computers, it can not be used in a different brand until it has been formatted for this new machine. Formatting the disc erases all the information

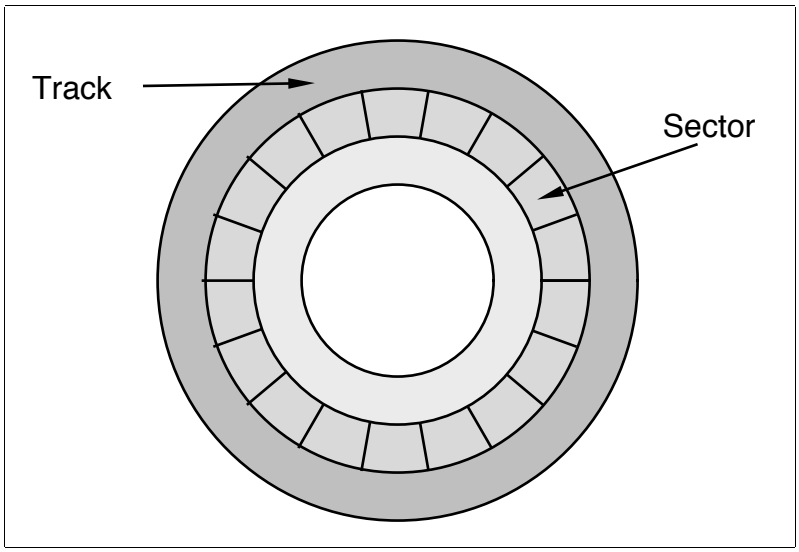


Figure 5.1: Tracks and Sectors on a disc

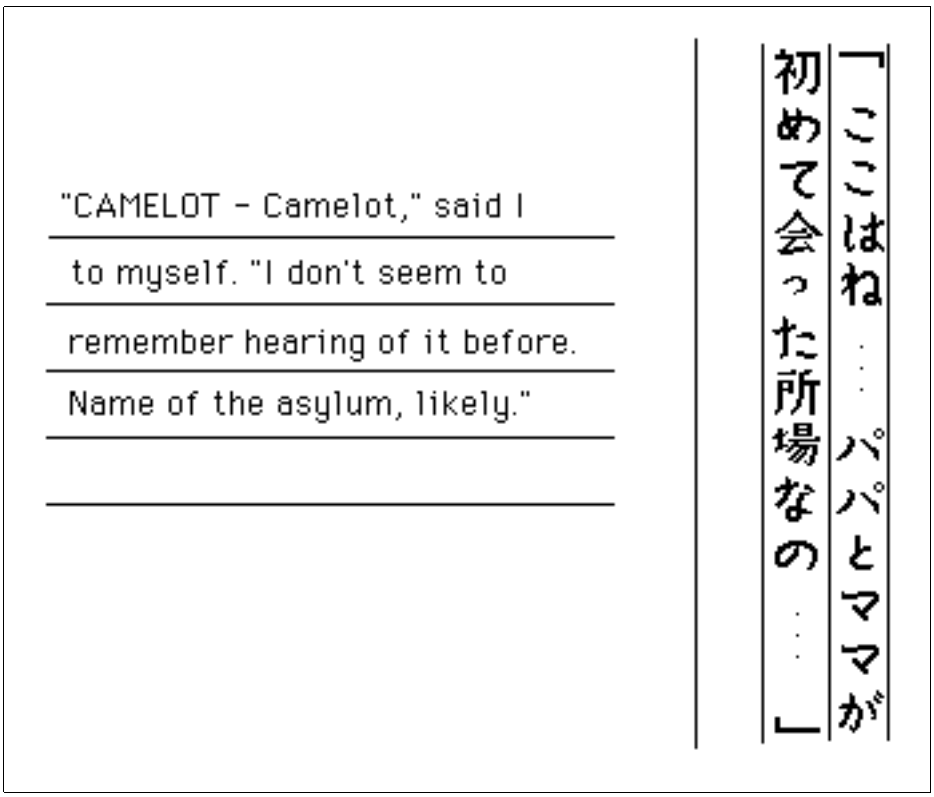
that is currently stored there.

You can think of formatting (or Initializing) a disc as drawing lines on a blank sheet of paper to act as guides for the writing that will come later. For English we write the lines horizontally, and expect to write on each page left to right and then down to the next line. For Japanese we write the lines vertically, and expect to write on each page top to bottom and then left to the next line. It would be very difficult to write on the wrong style of paper as Figure 5.2 shows.

Some companies are currently manufacturing drives that will read discs initialized for many brands of computers, so the problem of discs only being readable on certain machines should soon disappear.

5.2 Files

When we store information onto a floppy disc we store it in files which take up a certain number of bytes on the disc. When we store music on a cassette tape, we store sounds in songs which take up a certain number of minutes. There are two main types of files. Files can be **applications**, or they can be **documents**. Applications are programs like a word processor or a spreadsheet. They do things. Documents are files that are created by programs. You may create a psychology term paper (document) with



The beginning of
A Connecticut Yankee in King Arthur's Court
by Mark Twain

The ending of
Maison Ikkoku
by Rumiko Takahashi

Figure 5.2: Formatting a Piece of Paper

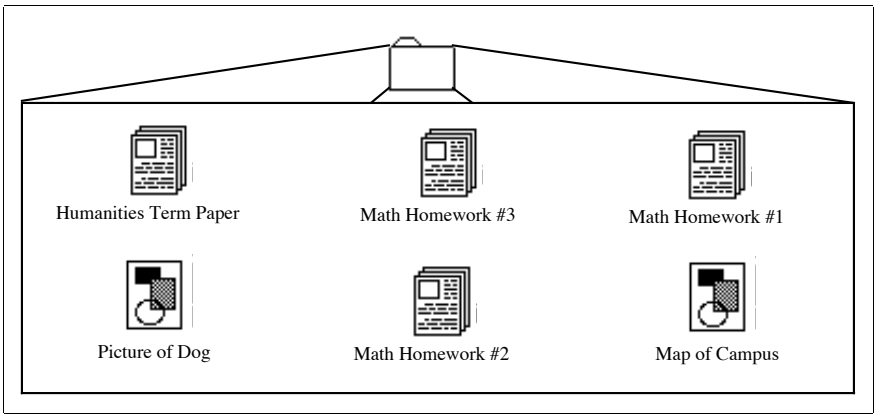


Figure 5.3: Without a Hierarchical File Structure

your word processor (application.) You can store several applications and documents on a floppy disc, and a lot of applications and documents on a hard drive. What is needed is a convenient way to organize these files.

5.3 Hierarchical File Structure

The current way of organizing files on a disc is through the use of a hierarchical file structure. When you use a file cabinet, you tend to group like papers together in a folder. You may have a folder called "tax info" or "Winter '90 papers." These folders allow you to organize your file cabinet. If you really get into the organization you can put folders inside other folders to further subdivide your categories. This same system is used by computers.

5.3.1 Directories

Computers store files in a cluster within a directory. A directory stored within another directory is called a **subdirectory**. A directory performs the same function as a folder in a file cabinet. For example, Figure 5.3 shows how files would be stored without a hierarchical file structure.

Instead of keeping these all in one directory it would be better to divide them up according to their subject matter. In this case we create two subdirectories called "Graphics" and "Text." In this case we put another subdirectory within the text-subdirectory to divide the text files into those for our Math class and those for our

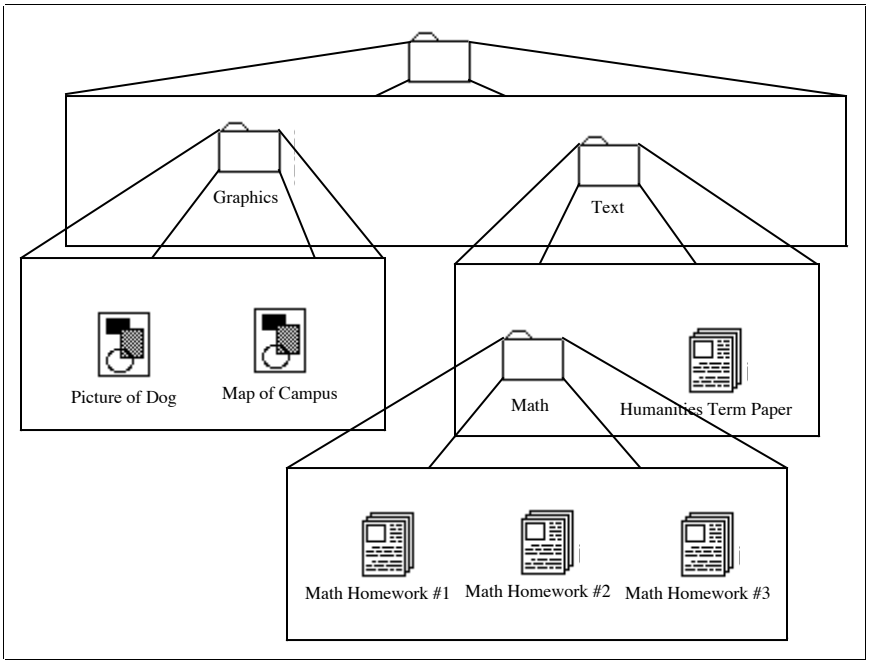


Figure 5.4: With a Hierarchical File Structure

Humanities class. Figure 5.4 shows how the files would be grouped.

If you have ever tried to find something in someone else's file cabinet, you know that each person has a unique way of organizing information. Directories allow each person to organize their files in a way that is most comfortable to them.

5.4 DOS Commands

Currently there are two major types of DOS available. One kind uses a command line interface (UNIX, and MS-DOS) while others use a graphical user interface (Finder, OS-2, Workbench, and Windows.) Both types have their advantages, but the graphical versions are becoming much more popular as files are displayed using icons, and you do not have to remember what commands you need to type to manipulate the files. Whichever type of DOS we use, there are certain commands that we are likely to need.

For most purposes we will need only need the following:

- Run an application

- Copy a file
- Rename a file
- Delete a file
- Create a directory
- Delete a directory
- Display a directory
- Go to another directory

5.5 DOS and You

Now that we have talked about hardware and software, the interface and DOS, we can finally describe what happens when you turn on a computer. This process is referred to as "booting up" the machine. This phrase comes from "pulling yourself up by your own bootstraps" since the computer is able to start itself up after you turn on the power switch.

As soon as you turn on the power switch the personal computer will look in its ROM to see what it is supposed to do first. Usually the machine will do a quick diagnostic check to make sure all its systems are in working order. Then it will look for a hard disc and if it can't find one it will look for a floppy disc in the disc drive and read some of the DOS into RAM. The screen will then show you that the machine awaits your command. At this point you can go about your business.

Now lets say we put a floppy disc into a Macintosh running the Finder (the Mac's graphical interface,) and put a floppy disc into an IBM-pc running MS- DOS (the IBM's command line interface.) When we boot-up the machines (turn the machines on) we see Figure 5.5:

Both of the machines are currently waiting for instructions, but I think you will agree that the graphical user interface shown on top looks much more friendly than the command line interface shown beneath it. Up until 1984 all personal computers looked like the bottom screen when you booted them up. A lot of black screen, a small prompt, and a blinking cursor. You can see why all newer personal computers come with a graphical user interface.

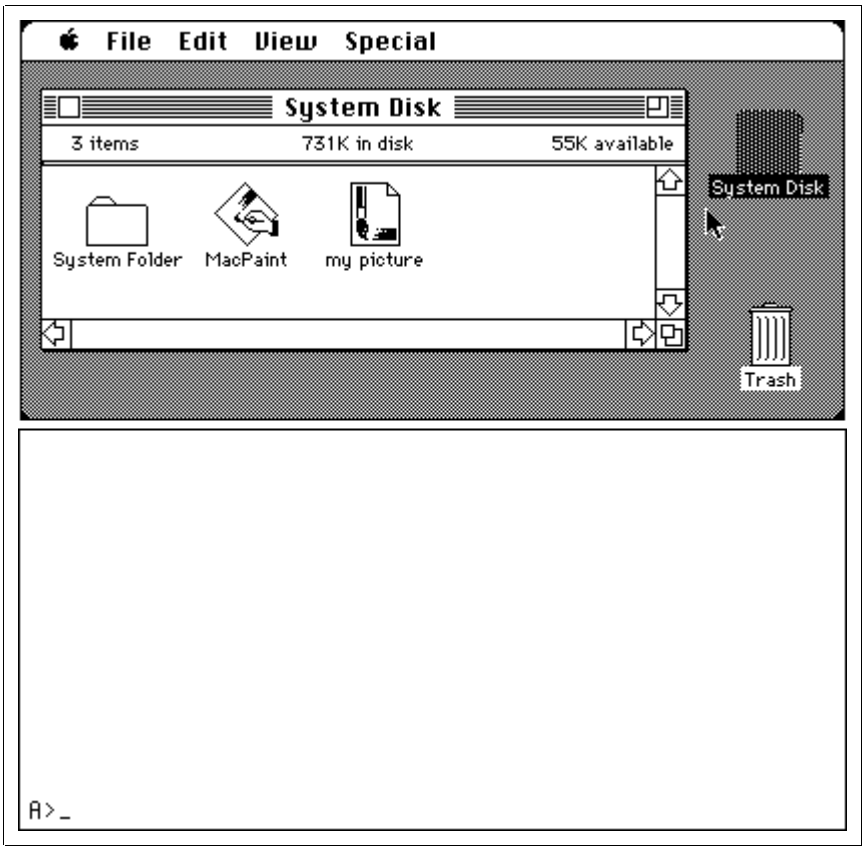


Figure 5.5: Boot Up Screens

The DOS is simply a large program that helps us manage the files (both applications and documents) on our discs. It acts as our home base when we use the computer. This is where we start after we boot the computer, and it is where we will return to after we have finished running an application.

5.6 Questions

1. What happens when you format a disc?
2. What is the difference between an application and a document?
3. Hierarchical file structures are used to organize:
   (a) your computer hardware
   (b) files on a disc
   (c) 'c' is for Cookie. That's good enough for me.
   (d) the RAM and ROM
4. What does formatting/initializing a disc NOT do?
   (a) Erase the disk
   (b) Rename all previous directories to 'untitled'
   (c) Make all previous files on the disc inaccesable
   (d) Reorganize the entire disc to store information on the disc for the first time.
5. What is a DOS and what does it do?

Chapter 6

Word Processors

Learning how to use a word processor is the most useful thing you will do in this class. A word processing program turns your computer into a very fancy typewriter with a lot of extras. With a typewriter you hit a key, and a letter appears on your piece of paper. With a word processor you hit a key and a letter appears on the screen. This allows you to go back later and make changes on the screen. Only when you have the paper exactly the way you want it do you send it to the printer to get a copy of it on paper.

6.1 Typing VS Word Processing

As the pen before it, the typewriter allows us to convert our thoughts into printed material for others to read. The typewriter is a general improvement over the pen as readable text can be created much faster. One of the major problems human beings have with typewriters is that we make make mistakes or we change our minds frequenty. Many of us prefer to use pencils over pens because we can erase our mistakes and correct them. With a typewriter it is not so easy to correct mistakes.

When we type a paper we tap away on the keyboard until a little bell sounds telling us that we are running out of space on the current line. We then must hit the carriage return key to tell the platen to rotate and move us back to the left margin on the next line down. We then repeat this procedure until we are finished. If we want to move back up the page we need to turn the knob on the side of the platen. If we need to work on a different page we must take out the current page and put in the one we want to work on, and hope we line up the page correctly.

Typewriters enjoyed a century without competition until word processors appeared on the scene. Today its almost impossible to find a typewriter without some features of a word processor built into it. Word processors allow us to have more more control over the document that we are preparing. As we type on the keyboard letters are displayed on the screen. The entire 'paper' is stored in memory. The computer allows us to scroll through the pages of this document on the screen and make changes to it: adding in a paragraph here, taking out a word there, and these changes are immediately reflected in the 'paper' shown on the screen. Being able to make changes before you get a print out is a tremendous advantage. If you find you have made a spelling mistake, you only need to go back and retype that word, not the entire page. Word processors allow you to move blocks of text from one place to another. If you think a paragraph would be better if it was moved back a couple of pages, you can easily move it without retyping it. I'm sure you've been asked to type a paper that's 5 pages long, and when you are finished typing it is only 4 and a half pages long. With a word processor you can go back and push the margins in until the paper becomes 5 pages long.

Most word processors today are **WYSIWYG** ("Wizzywig") which stands for "What You See Is What You Get." That means that what you see on the monitor screen is 'exactly' the way the text will look on the page when you finally print it out. There is no guesswork involved so word processors save you time and paper. You also have the ability to store your paper on disc. No need to make carbon copies anymore. You can print out another copy any time you want to.Figure 6.1 shows the screen of a typical word processor.

6.2 Features

Word Processors offer you a lot of extra features. Most people are very bad at spelling, but computers are pretty good at it, so most word processing programs have a spell checker built in. You tell the computer to check your paper and it will go through the paper comparing each word in your paper to the words in its built in dictionary. If it can't find a word it will ask you if that word is spelled correctly, and possibly give you some alternate (correct) spellings for the word. More advanced word processors have thesauri which will allow you to pick more impressive sounding words. Others have grammar checkers to make sure that your grammar is correct. Some Word processors come with special legal or medical dictionaries as well as those words for everyday use. A common computerized dictionary will have 50,000 words in it, and most let

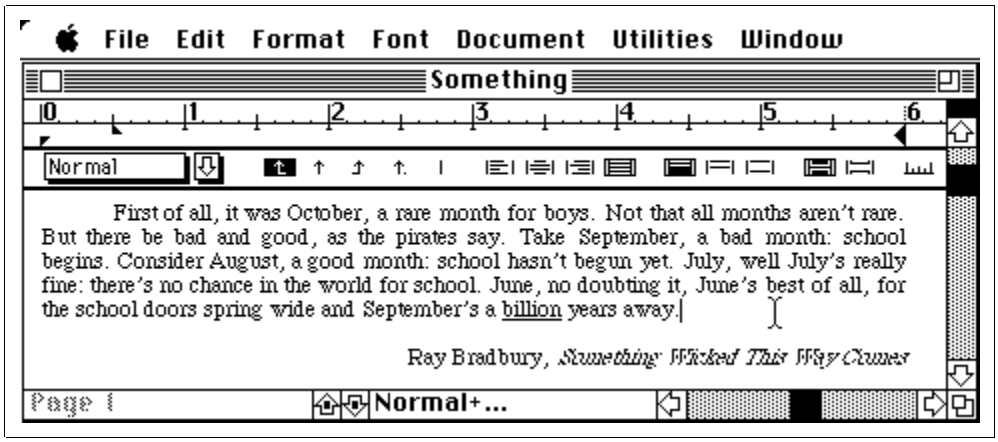


Figure 6.1: Typical Word Processor

you add your own words.

6.3 Options

With a typewriter you can make words all capitals, or go back and underline, or type them again for emphasis. A word processor gives you these options and more. They allow you to type in different colours, fonts, font sizes, and styles. They allow you to set the spacing between the lines and what kind of justification the paragraphs will have. You can type in a header once and have it automatically put on top of every page. You can have indices and tables of contents generated automatically. All of these are shown on the screen of a WYSIWYG system. All of these options can be changed whenever you ask for them to be changed, even after you have finished typing. The combined effect of all these options allows you to create documents like this text. Several of these options are shown in Figure 6.2

6.4 Desktop Publishing

A "buzzword" of the late 80's was Desktop Publishing. Desktop Publishing is the use of a personal computer to produce professional looking documents. With this in mind, most word processors allow you to insert illustrations or diagrams into your text, giving you the power of a small print shop. Many magazines, newspapers, and

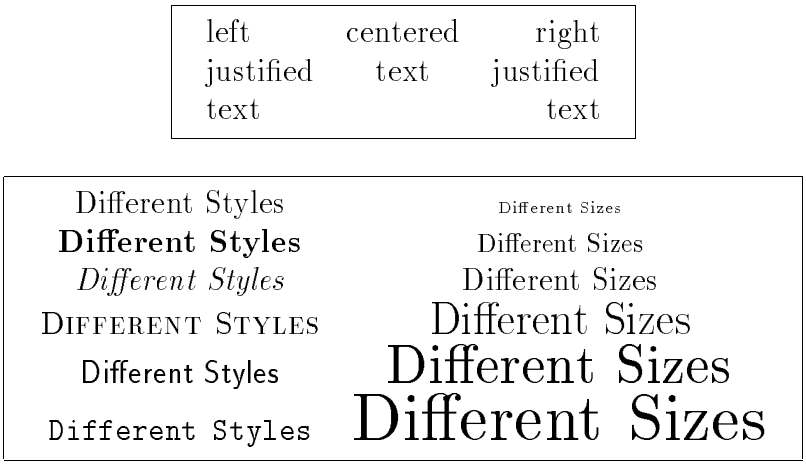


Figure 6.2: Options

publishing houses now use computers to do their layout work, and in some cases to print out the final product. The desktop publishing market was given a great boost by the introduction of low-cost Laser Printers in 1985.

In 'ye olden days' if you wanted to combine some printed material and a diagram you would use a typewriter to type up your words on one sheet of paper (where you hope tyou don't mistype anything) and draw your diagram on another sheet of paper. Then you cut out the diagram with a pair of scissors, and get some celophane tape and tape the diagram onto the page with the text. You try to align it correctly and tape it down flat. You then feed this Frankenstein's monster of a document into the photocopier and hope it doesn't jam. Finally you receive a page with blurry text and celophane tape marks clearly visible around the diagram. O well, you were living in the 70s, so you didn't expect much. Today you can produce this entire document on a personal computer. You type in the text using a word processor, and draw the diagram with a drawing program. You use the computer to merge the two documents and print out the final document on a laser printer. Figure 6.3 shows the difference.

As you can see, there are a lot of bonuses with word processors. The fancier you need to get, the more useful a word processor is to you, but the converse is also true. If you are going to write a grocery list, then you should use a pencil and a piece of paper. If you are going to fill out a tax form, then use a pen or a typewriter. Again, it is important to use the correct tool for the job.

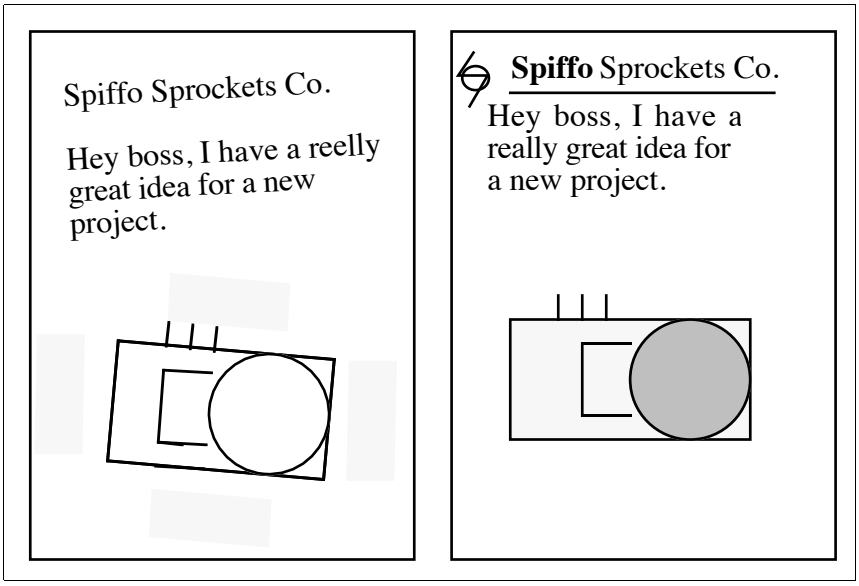


Figure 6.3: 70s memo VS 90s memo

A Word Processor is not the only tool available on a computer for manipulating text. There is also the **Text Editor** and the **Document Processor**. A text editor is a word processor without a lot of the special features. Text editors are mostly used to write computer programs and create files for use with document processors. While word processors are good for writing papers, they do not have enough features for writing long papers or textbooks. Document processors are good for writing long papers and textbooks. The names of the three products suggest their power: Text Editor, Word Processor, Document Processor. A processor is more powerful than an editor and processing a document takes more power than processing a few words. Figure 6.4 shows this in a pictorial form.

6.5 Text Editors

Text editors are usually used for writing computer programs. Computer programs do not require fancy fonts, or fancy page layout options. The word processors of the early 80's had the same power as today's text editors. Text editors are somewhere in between typewriters and word processors.

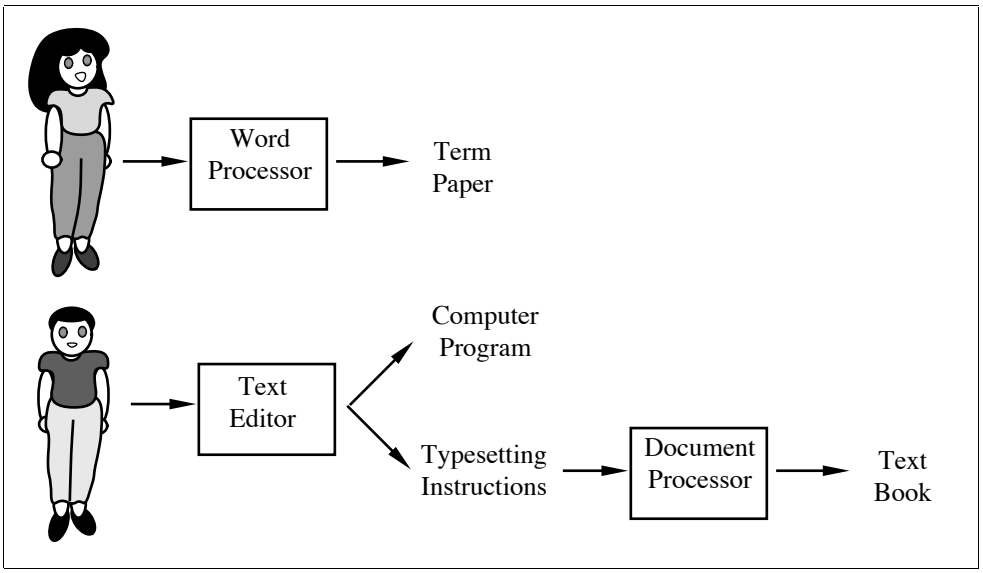


Figure 6.4: Comparison of Text Manipulation Programs

6.6 Document Processors

Document processors are different from text editors and word processors in that the user does not use a document processor directly. A document processor acts like a human typesetter. You tell him what you want typeset, and you give him instructions as to how it should look, and then you leave him alone to do the job. That is how document processors work. The user will commonly use a text editor to type in a file of information containing the words to be typeset as well as instructions as to how to typeset them. This file is then sent through the document processor which returns to you the typeset version ready for printing.

Why would we want to use a document processor over a word processor? The key is generality. With a word processor you must manually set up the style for your paper. If you need to change that style you must go in and change it yourself. With a document processor you simply change the command that set up the style. For example, lets say you want to submit your article on Elvis being a Martian to Scientific American. Scientific American expects a very specific style. After they reject your article, you may want to try the Weekly World News. You will have to change the article to fit the new style. With a word processor you may have to go in and adjust the style of each paragraph independently. With a document processor it can be as

simple as changing the command {.style "SCIAMER"} to {.style "WWNEWS"} and that's it. The document processor will change all the paragraphs for you. The big trend today is to enhance word processors to give them the flexibility of document processors, but keeping their WYSIWYG interface.

6.7 Comparison

Figure 6.5 compares some of the basic features of Word Processors, Text Editors and Document Processors. The availability of some of the features depend on the specific software being used, so this table should only be used as a general guide.

6.8 Word Processors and You

So what can you really do with a word processor?

- Write term papers and lab reports
- Write letters (especially form letters)
- Write your resume
- Write notices with huge letters

How about a text editor or a document processor? Well, if you are a CS major you will use a text editor a lot when you write computer programs. You will probably use a document processor to write your thesis or dissertation, or any textbooks you feel inspired to write. If you are not a CS major, and do not plan on being a published author, then you will probably not have a need to use these other two products.

After all this hype you might be ready to chuck the ol' typewriter right out the window. Well don't. First of all it might hit somebody on the head and injure them severely, and second of all you will still need it. When you learned to type you didn't throw away your pencil. It is still a very useful device. The same is true with a typewriter. You will still need a typewriter to do the following things:

- Type an address on an envelope
- Type information into forms

Criteria	Word Proc.	Text Editor	Document Proc.
Cut, Copy, Paste	Yes	Yes	No
Search, Replace	Yes	Yes	No
Center, Justify	Yes	No	Yes
Indent, Tab	Yes	Yes	Yes
Bold, Underline Italics, Subscript, Superscript	Yes	No	Yes
Print previewing	Yes	No	Yes
Thesaurus Spell Check	Yes	No	No
Footnote Endnote	Yes	No	Yes
Inclusion of lllustrations	Yes	No	Yes
Programming language formatting	No	Yes	Yes
Consistent and automatic formatting	Some	No	Yes
WYSIWYG	Mostly	No	No
Programmable	Macros only	Macros, limited editor commands	Completely programmable
Good uses	Composing letters, short term papers	Writing programs	Composing long term papers, textbooks
Popular titles	Word Star, Word Perfect, MacWrite, Microsoft Word	VI, Emacs	NROFF, Scribe, TEX, LATEX

Figure 6.5: Comparison

6.9 Questions

1. What does WYSIWYG stand for?
2. What are the different primary uses for a word processor, a text editor, and a document processor? Be specific.
3. Desktop Publishing is:
   (a) running a word processor on a GUI computer
   (b) posting messages from your computer to a BBS
   (c) using a personal computer to produce professional looking documents
   (d) not possible using a personal computer
4. Why is WYSIWYG good to have in a word processor?
   (a) It allows you to see what will be printed before we actually print
   (b) It allows full justification of paragraphs
   (c) It ensures correct spelling in a document
   (d) It ensures you always have a backup copy of your document

Chapter 7

Telecommunications

We have said before that different brands of computers have different CPU chips and different ways of storing information on disc. This makes it difficult for different brands of computers to exchange information. One of the ways computers can exchange information is over the telephone lines using a **modem** (short for MODulator DEModulator.) Since the information is encoded into sounds for transmission over the phone lines, different brands of computers can 'listen' to that sound and translate it into their own format. This allows users of different computers to send each other **electronic mail**. Computer users can also use **electronic bulletin board systems** to talk with people all over the world.

If you want your computer to be able to talk to other machines, you need a modem. The modem attaches to your computer, and to a standard phone line. The modem gives the computer the same ability you get with a telephone. The computer can dial phone numbers, it can send information, or it can receive information. Your computer can then talk to anyone else's computer which is also hooked up to a phone line as shown in Figure 7.1.

7.1 Protocol

Different computers have different CPUs and different operating systems etc. They are like people. We could just walk around grunting and growling at each other, but that would not help us advance civilization. We are able to convey ideas to each other because we have settled upon certain conventions. We call them languages. Computers call them Protocol. In order for you to communicate with a friend you

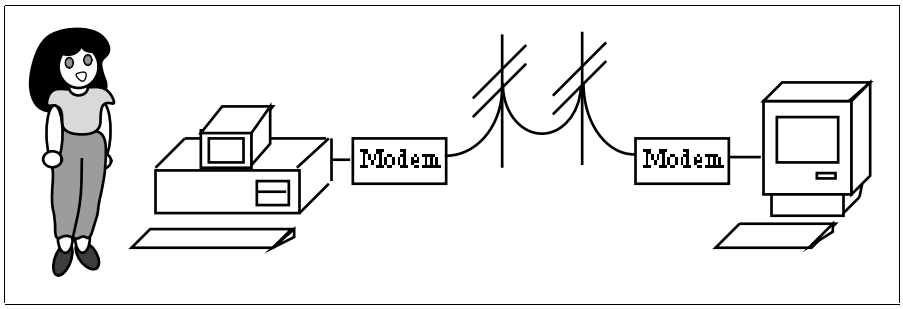


Figure 7.1: Telecommunications

must decide on a common language. In order for your computer to communicate with another computer, you must decide on a common protocol that their modems will use. This way the computer receiving the information knows how the sending computer converted the information into sounds, so it can be converted back. Without a common protocol the computers will simply be sending meaningless sounds back and forth.

7.2 BBSs

When you walk through the halls of Wayne State, or walk into a grocery store you will see bulletin boards on the walls. People use these bulletin to post announcements or leave messages for people. The telecommunications equivalent is called, surprisingly enough, a "bulletin board." Using your modem, you call up a bulletin board system, or BBS for short. There you can read public messages left by other people, leave public messages of your own, send private mail to other users, and hold conversations with people you have never seen nor heard. The first personal computer BBS was written in 1978 by Ward Christensen and Randy Suess, and was called CBBS.

You would actually be surprised to know how many small bulletin boards are out there today. Its really incredible. Within the Detroit area there are almost 400, and some of them have been running for almost a decade. Most of these are free to use, except you have to pay the phone bill if its not a local call. Calling BBSs can be very addictive. I have known people who ran up $500 phone bills calling different BBSs all over the country. There are a lot of people out there to talk to, and a lot of topics to talk about.

BBSs are usually divided into three areas: Public Messages, Private Mail, and a

File Section. In the public message areas people leave messages for others to read. This is where all the discussions go on. You can leave a message, or respond to someone else's message publicly. All of the other readers can read your message when they call the BBS again. You may want to leave another user private mail, so only she can read it. You would leave private mail in the private mail section. Most BBSs also allow their users to leave messages anonymously. The File Sections contain various programs and BBS phone numbers. The bulletin board program keeps track of the messages and the users. All of this information is stored on disc at the BBS computer, so usually the BBS computer has a large hard drive.

The owner, and manager of the bulletin board is called the **sysop**, short for system operator. The sysop decides what kind of system to run and what rules the users will have to obey. Many small BBSs are computer specific, that is only Amiga users would find things of interest on an Amiga bulletin board. Users of these systems talk about new software and hardware that is available for their machines. Used software and hardware is bought and sold. These computer specific boards are also good places to get hold of (free) public domain software for your computer. Many small BBSs are not computer specific. Their users discuss a wide range of topics not necessarily computer related (e.g. Star Trek, politics, books, ham radio etc.) These BBSs tend to be much more interesting. Very little actual "computing" is done on either type of BBS.

All you need to set up and run a small bulletin board is a personal computer, a modem, a phone line, and bulletin board program. The computer running the BBS will be running its BBS program all the time, waiting for someone to call. As soon as the phone rings, the computer will answer the phone and begin talking with the computer on the other end. If a human being attempts to call a BBS, he will hear a very loud beeeeeeeeep and that's about all. Usually people put in a second phone line for the BBS, so they have one phone line for the BBS users to use and one for their human friends to use. As with a regular phone, only one person can call at a time.

Now BBSs aren't listed in the phone book, so how do you find out what their phone numbers are? Well, you could just randomly call phone numbers and see if a computer answers the phone, but that's not very polite. A better way is to ask at a local computer store. The people there can usually give you a couple of numbers. Each BBS will usually list the numbers for several other BBSs, and very soon you will find a BBS with a very large list of other BBS phone numbers, and then you're all set.

7.3 BBS Lingo

Like CB radio users (remember them from the late 70's?) BBS users have their own lingo that they use to enhance conversation.

For example a normal conversation on a BBS would look this this. BUT IF YOU WERE REALLY, REALLY ANGRY YOU WOULD TYPE IN ALL CAPS LIKE THIS TO SIMULATE SHOUTING! You often find people typing in all caps when they are 'flaming' another user of the BBS. When you 'flame' someone, you insult them.
Of course there are more acronyms to 'simplify' conversation:
BTW - By the way
IMHO - In My Humble Opinion
IOTTMCO - Intuitively Obvious To The Most Casual Observer
RSN - Real Soon Now
RTFM - Read the F*ck*ng Manual
WRT - With Respect To

There is also a large menagerie of faces that you can make. Faces are used because typed words have no tone of voice, so others may not be able to tell whether you are being sarcastic, or serious. Faces help clarify your feelings on the subject.

```
:) - happy face, or smiley. Used to indicate humour.
:( - frowning face. Used to indicate sadness or displeasure.
;) - winking face.
etc.
```

BTW, the Japanese draw their faces this way:

```
^_^
```

7.4 Terminal Program

In order to communicate with these electronic bulletin boards you need four things: A phone line, a computer, a modem, and a terminal program. The first three are the necessary hardware, and the terminal program is the necessary software. Just as a word processor turns your computer into a fancy typewriter, a terminal program

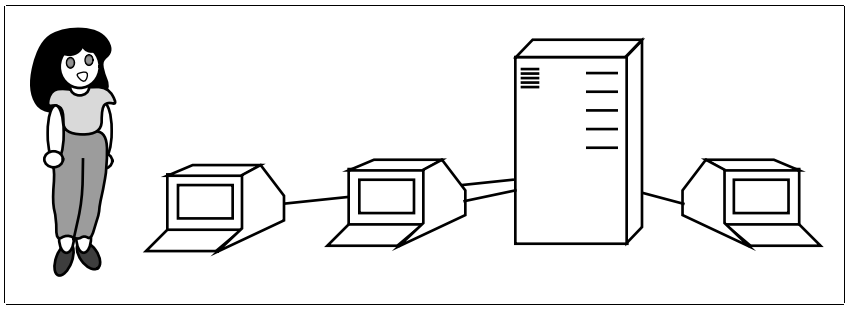


Figure 7.2: Terminals

turns your computer into a fancy telephone. Using the terminal program you give commands to your modem. You set the protocol you need, and the phone number you want to call and the modem will make the connection. Usually when you buy a modem it comes with a free terminal program.

Back when the only computers were mainframes, the mainframe would be locked away in a special room. The users communicated with the mainframe using a terminal as shown in Figure 7.2. A typical terminal would have a keyboard and a monitor, and a small circuit board, but no disc drives. Its only function was to allow a person to communicate with the mainframe. The name has stayed with us, so now when you call a BBS, you are turning your personal computer into a terminal. You use it only as a gateway to this other computer. Just as there are many different brands and models of personal computers there were/are many different brands and models of terminals. A terminal program allows you to choose which type of terminal you wish to emulate.

As well as calling BBS systems, you can also use your modem to call some Mainframe computers. Wayne State is part of MTS (the Michigan Terminal System.) You can call up Wayne State and access computer accounts on the local Amdahl mainframe, or the mainframe computers at other Michigan universities. Where BBSs are usually used for BS'ing, mainframe computers are used for computing. Mainframes will typically have several hundred users calling in and working at the same time. To each user, it seems that he is alone on the machine. The mainframe gives each user in rotation a small slice of time working with the CPU, but mainframes are so fast that you can not notice any delay.

7.5 Passwords

Bulletin boards have more than one user (or it would get pretty lonely,) so each user has her own mailbox for receiving private messages. It would not be good for other people to be able to read your mail without your approval. For this reason each user is assigned a unique identification code, and a secret password. Your identification code is public knowledge ... like your name and phone number are listed in the phone book. The password allows you to verify your identity. Each computer system you belong to will assign you an identification for that system. Each user is given an identification code for the same reason the government gives you a social security number. There may be two users of this BBS with the same name, but each will have his own unique identification code.

Mainframes also make use of identification codes and passwords. Each of the mainframe's many users is allocated a certain amount of disc space to use. It would be rather annoying to have other people reading your files without your permission. Like a BBS, a mainframe provides the capability for the various users to send messages to each other. Mainframes also allow many users to have access to a very large, fast, and powerful computer to do their work on.

Personal computers do not usually employ a system of passwords. It is assumed that either only one person uses the computer, or that each person keeps his own files on his own floppy disc. Some public personal computers do have security measures so that only appropriate users can access certain files on the Hard Drive.

7.6 Information Services

As well as thousands of small bulletin boards spread across the country, there are a few large national bulletin boards, though they tend to give themselves more impressive sounding names such as information services. These larger bulletin boards tend to charge you a certain amount of money per hour you use them, in addition to the money you pay your local phone company for the call. They are usually divided into sections for various computers, and various general topics of discussion. Small bulletin boards are usually run on personal computers and can usually only handle one caller at a time. The larger systems run on mainframes and can handle hundreds of users simultaneously. One of the oldest services, CompuServe (CIS), started operation in August of 1979 and now has 500,000 users. GEnie started up in October of 1985 and now has 200,000 users. Recently IBM's Prodigy system has been getting a lot of

hype. It is one of the newer information systems to appear.

7.7 E-Mail

Electronic mail (E-mail) is a lot like regular US-Mail. You type in your message, address it to a certain other user, and then send it off. Everything is electronic ...there is no paperwork. Some computers are connected via phone **networks** so that you can send mail to other computer systems in other states and other countries. These networks allow you to send E-mail to a local computer but address the message to a user of a computer far away. The computer will then send your message through the network of other machines until it reaches its destination. The response time for electronic mail is substantially faster than that for the US Mail (actually the response time for an armadillo with a backpack is faster than the US Mail.) Electronic mail can usually be delivered within minutes.

Each time a user calls (or "logs in") to a BBS or a mainframe, he is told whether there is any new mail waiting for him. He can then choose to read this mail, or read it later. When he does finally read it he may want to store the message on his disc, or reply to the person who sent the message. In effect, its just like regular paper mail being sent to a post office box, except that the computer handles all the work.

7.8 Uploading & Downloading

Just as we can move messages from place to place without using paper, we can move programs from place to place without using discs. Usually we get a new program by buying it on disc at a store that sells software. Public Domain software is not sold in stores. It is usually distributed electronically. Most BBSs have file sections where you can download or upload programs. When you download a program you make a copy of the program that is stored on disc at the BBS, and put it on one of your floppy discs. The program is sent through the phone lines from the BBS to your computer. Uploading is just the opposite. You take a program stored on one of your floppy discs and put a copy of it on the BBS's disc, so others will have access to it. Figure 7.3 illustrates.

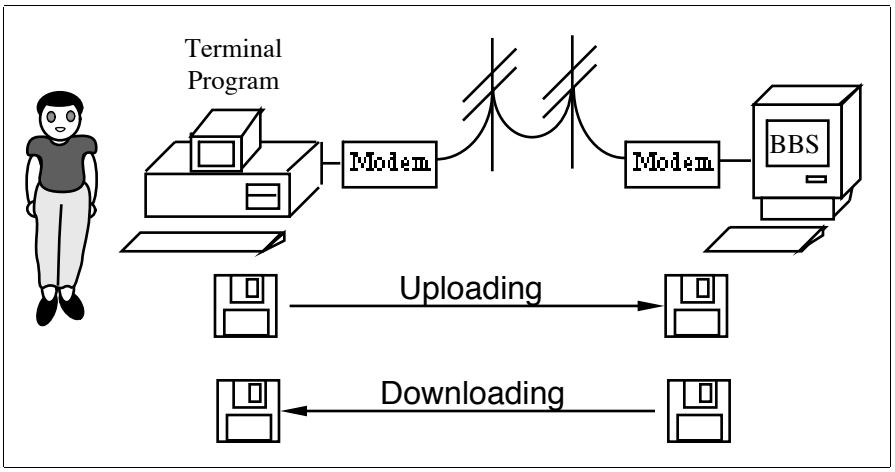


Figure 7.3: Uploading and Downloading

7.9 Hackers

As with all things there is a good side and a bad side. Allowing people to communicate freely with these systems on a wide range of topics is certainly good. Of course there are some people who wish to take advantage of this. Over the last few years there has been an interest in "hacking" in the media. A hacker is someone who really gets to know the nitty-gritty details of a system and pushes it to its limits (for an excellent discussion of hackers see *Hackers - Heroes of the Computer Revolution* by Steven Levy.) Now this term has gotten a very bad connotation. Today hackers are people (usually around 13 years old) who gain unauthorized access to computer systems (usually by pretending to be one of its users.) Some hackers are benign and only want to look around. Some are clumsy and destroy data by accident. Some are malicious and intentionally destroy information. While this is not a big deal with a local BBS, it can be if the computer system is for a hospital, or NASA.

Mainframe computers are like apartment buildings. Each tenant leases some space to use. Each tenant has their own space which is separate from everyone else's. Each tenant has a key (password) which allows him to get into his space, and prevents others from getting into his space. Now each apartment building (mainframe) also has a manager (system operator). This manager is responsible for keeping the tenants (users) happy, and keeping the apartment secure. Someone could walk into your apartment building and go to various apartments twisting the doorknobs to see if the rooms are locked. If the door is unlocked, he could go in and look around ...maybe

search through some closets and drawers and then leave. Now this person hasn't done any damage, but I don't think you would be too happy if you were the tenant. The word 'trespassing' comes to mind. That is what most hackers are doing. Now there are some that go even further. Why go around rattling doorknobs when you could steal the managers master keyring. Then you could go into any room you wanted for whatever reason. If a hacker can get a hold of the mainframe's master password list he would have that ability.

Currently most computer systems are very open. Their purpose is to allow people to share information easily. In his book "The Cuckoo's Egg" Clifford Stohl compares them to a small town where everybody still leaves their door's unlocked at night. It is perhaps a little naive in this day and age. Adding on massive security systems makes it harder for people to communicate and in many ways defeats the purpose of these systems. What hackers do is betray the level of trust that keeps the systems running smoothly.

Of course the government has become interested in hacking with the large numbers of computers that the military uses. The FBI and the Secret Service take a great interest in hackers. Unfortunately these agencies have a tendency to overreact quite a bit on their seizures ...entering with an abundance of firepower and few warrants, and then proceeding to carry off everything in sight. There are stories, not necessarily apocryphal, about police agencies dusting floppy discs for fingerprints. Complicating the matter is that there are very few legal rulings concerning E-mail, and bulletin boards and how they relate to Amendments 1, 4, and 14 of the Constitution.

7.10 Fone Phreaks

Another group of people you might have heard about are Fone Phreaks. Just as hackers are interested in computer systems, fone phreaks are interested in communications systems. Unfortunately, many are also interested in credit card fraud and other nasty things. Ma Bell was the obvious target in the mid 70's. Even Steve Jobs and Steve Wozniak - future founders of Apple Computer Corp. huckstered equipment to make free long distance phone calls. When other long distance phone companies were becoming popular in the 80's each user was given an identification code to use when making calls. If someone else got your code they could make calls to anywhere they wanted and charge them to you. Typically fone phreaks would charge their calls to large corporations, and play tag with the FCC, much as the software pirates play tag with the FBI.

7.11 Security

You can never be completely secure, but you can be safer if you choose a good password. Bad passwords include your name, or HELP, or SAMPLE, or 123. Your password should be something that you can remember without writing down, but something that would not be obvious to a casual acquaintance. Other passwords to avoid are names of girlfriends/boyfriends, unless the relationship is really stable. Often couples will break up and then you want to change your password, or your former companion will try to get back at you by erasing all your files.

There are several ways that computers can be made more secure. One way is to have a call-back modem. In this case, you call the computer, hang up, and then the computer calls you back at a preassigned phone number. Its even safer if you do not allow people to call into your computer from outside your building. This way you only have direct connections between computers. The highest level comes from forcing the person to go to the computer itself, and not connecting the computer up to anything else. The trade off is security versus convenience. The more convenient something is, the less secure it is.

7.12 Viruses

We said that you can often download software from BBSs. It would be nice to assume that this software is safe to download. Unfortunately this is not true. Some software contains computer viruses. A computer virus is like a molecular virus in real life - it spreads and infects others. Computer viruses are small programs hidden within an application such as a word processor. Running an application infected with a virus first activates the virus and it looks around for other programs to infect on other discs currently in the computer. Then the 'real' program runs so you are unaware that anything else has happened. The virus then spreads from disc to disc. As in real life some viruses are harmless, others can cause great damage. Some times the person who wrote the virus was clumsy and his poorly written virus causes unintentional side effects.

Now you might think you would be safe if you only use store-bought software, but its not true. Even computer companies can get viruses, and then an official packaged product sent out by the manufacturer can inadvertently contain a virus. You could protect yourself from viruses by never downing software from a BBS, and never copying any programs from your friends, and never exchanging any data with

anyone else ... but then your computer loses a lot of its worth. Just as hackers have interfered with the trust that mainframe users had, viruses have made personal computer users less trusting, less open, less willing to share information.

Viruses tend to act in predictable ways, so there are software products out there to make sure that your software doesn't do anything suspicious. These programs stop viruses from spreading. Another group of disinfectant programs will search through the applications on your disc looking for hidden viruses. If a virus is found it is removed, and then the application will work normally again. With software available on worldwide computer networks, it is possible for a computer virus to spread around the world within hours. Updated disinfectant programs to handle this new virus appear about a week later.

7.13 Worms

Worms are to networks of computers what viruses are to personal computers. A worm is a piece of software designed to spread through telecommunication networks, reproducing itself as it goes. Worms do not hide within other programs to sneak in to your computer, they just crash through the front door, or the back door, or through the cracks in the floor. As soon as they get in, they try to spread to as many other computers as possible.

7.14 Logic Bombs

A logic bomb is like a timed explosive device hidden within a program. When a certain date rolls around or a certain program is run "something" will happen. This something may be a message printed on the screen saying "Dukakis for president", or it may be your hard drive erasing itself. Logic bombs are usually spread by viruses so they can annoy the greatest number of people possible.

7.15 Trojan Horses

A Trojan horse is a program written so that it appears to do one thing while secretly designed to do something else. A common Trojan horse program in a university setting will look like a terminal program. When you go to the lab and use this terminal program to call a BBS or log in to a mainframe, the terminal program will

appear to run normally. Secretly, the 'terminal program' is keeping track of your ID#, and password as you type them in. This way the author of the 'terminal program' can come by ocassionally and get a listing of valid ID#s and passwords to use in unscrupulous ways.

Are all these bad things worth worrying about? Yes and No. They are good to be aware of, but they are not worth losing any sleep over. Its like the threat of someone scraping their keys down the side of my car. I realize that it can happen, and I can protect myself a little, but there's not a whole lot I can do about it.

7.16 Telecommunications and You

Telecommunications technology has greatly improved the speed and amount of communication. The modem, like the telephone before it, gives you instant access to people around the world. BBS systems lets you talk with people about topics ranging from neural networks to last week's episode of "Twin Peaks." Important things get discussed as well as popular culture. It is no longer a secret and privileged medium. It has become simply another means for "average" people to communicate their ideas and opinions freely ... which is perhaps the greatest benefit that technology gives us.

7.17 Questions

1. What does a modem do?
2. What are passwords used for?
   (a) to identify yourself to a computer system
   (b) to verify your identity to a computer system
   (c) to read your mail on a computer system
   (d) to send mail on a computer system
3. A computer virus is a:
   (a) small program that hides within other programs
   (b) small program that hides within a computer's ROM
   (c) small program that spreads itself over computer networks

(d) small program that alters the tracks and sectors of a hard disc

4. What is a BBS, and what is it used for?

5. What equipment is needed to communicate with another computer using the telephone lines?

6. How can a mainframe allow many users on the system at one time?

   (a) a mainframe is actually many separate personal computers

   (b) each user is given a small 'slice' of time in rotation

   (c) all the others must wait until the first person signs off

   (d) no one uses mainframes anymore so only one person is on at a given time

Chapter 8

Databases

Computers are very good at storing information, and when you have a lot of information to store you should use a database. A database provides convenient and efficient ways for the user to access, add, modify, and remove information stored within it. The key words here are convenient and efficient. Depending on the amount and type of information you want stored a database may not be convenient or efficient. Your brain, or a piece of paper may be more convenient and efficient.

When personal computers were new, companies were trying to figure out why you "needed" to own one. One suggestion was that you "needed" a computer to store your recipes on. You could then throw away the cookbooks and life would be wonderful. Unfortunately this was not thought through very well. First of all you needed to have the computer in the kitchen, preferably near the oven and the sink - near hot sauces and free flowing water. Whenever you wanted to cook something you would have to go and turn the machine on, load up your recipe program, find the recipe and display it on the screen. While cooking you would need to run back and forth between the computer and your pots and pans, being VERY careful not to drop some egg-yolks into the keyboard, or smear flour onto the monitor screen. This was not convenient, not efficient, and not very smart.

Figure 8.1 compares three of the more popular ways of storing information. The human brain is a marvelous piece of equipment, but it has some limitations. When the brain could no longer store all the information we needed to know we began to store information on paper. When the piles of paper began to fill entire rooms we began to use computers. Each of the three ways of storing information has some advantages and some disadvantages.

Databases are good when you have a lot of data and it does not have to be

Criteria	Brain	Paper	Computer
Amount of info	a few	some	a lot
(phone numbers)	10-20	50-100	thousands
Recall speed	fast	good if organized	fast
Reliability	low	high	high
Convenience	right there with you	good if small	you go to it
Update	terrible	fair	good

Figure 8.1: Comparison of Data Storage Devices

portable. This means that a database is not very useful for the average person, unless you happen to be a serious collector of stamps or squished bugs and need to store a lot of data. Databases are useful for companies with lots of data to keep track of such as airlines, universities, hospitals, banks, and the IRS.

8.1 Terminology

There are three main types of databases: Relational, Network and Hierarchical. We will be concentrating on the **Relational Model** since it is very popular currently. The relational database stores its information in one or more relations. Another name for a **relation** is a **table**. There are several "database" products on the market for personal computers that are not really databases at all. The reason for this is that most personal computer users do not need a real database. Hence they give you a watered down database that fills your nonexistent needs. Here, we are going to talk about real databases since you may encounter them if you go to work where there is a lot of data being stored.

Teacher table

Name	OH Bldg	OH Room	OH Time
Biasu	Mack	123	11
Leigh	Main	106	10
Jekyll	North	234	3
Phibes	State	234	2
Johnson	State	405	4

Teaches table

Class	Name
671	Biasu
511	Jekyll
680	Phibes
441	Leigh
101	Johnson
871	Biasu
880	Phibes

Course table

Class	Class Bldg	Class Room	Class Time
680	State	129	4
511	North	102	3
671	Mack	123	11
441	Main	211	12
101	State	312	8
871	Mack	450	10
880	State	129	6

Figure 8.2: Information in Three Tables

8.2 Example

The database shown in Figure 8.2 contains three tables. Each table contains different types of information or fields. Another name for **field** is **attribute** or **column**. The Teacher Table has 4 fields: Name, OH Bldg, OH Room, and OH Time. Each table also contains records, or sets of data with values for the fields. Another name for **record** is **row** or **tuple**. The Teacher Table has five records - one for Professor Biasu, one for Mr. Leigh, one for Dr. Jekyll, one for Dr. Phibes, and one for Mr. Johnson.

Why do we have three tables, when we can combine all the information into one table as shown in Figure 8.3? Having three tables is better for the following reasons:

- reduces duplication
- increases security
- decreases search time

	Class	Name	OH Bldg	OH Room	OH Time	Class Bldg	Class Room	Class Time
Combined table	671	Biasu	Mack	123	11	Mack	123	11
	511	Jekyll	North	234	3	North	102	3
	680	Phibes	State	234	2	State	129	4
	441	Leigh	Main	106	10	Main	211	12
	101	Johnson	State	405	4	State	312	8
	871	Biasu	Mack	123	11	Mack	450	10
	880	Phibes	State	234	2	State	129	6

Figure 8.3: Information in One Table

- simplifies updates

Each separate table hold some specific information. The Teacher table holds information about certain teachers, their office, and their office hours. It only contains information about the teachers. The Course table holds information about certain courses, and where and when they meet. It only contains information about courses. The Teaches table allows the information in the other two tables to be combined. When we combine the information into one table we see things like Professor Biasu's and Dr. Phibes' office information is written twice. A teacher's office has nothing to do with what class(es) he is teaching.

It is faster to look through a smaller table than a larger one. It is also generally faster to combine a few small tables than to look through one big one.

What happens if Professor Biasu decides to move his office from one building to another. With the three separate tables we only need to look through the teacher table until we find Professor Biasu and change one entry. With the combined table we need to look through the entire table because we do not know how many times Professor Biasu's name will appear. This would be very wasteful in a big database, and could lead to errors if we forget to change one of the entries.

Security is very important with databases. With one big table either a person has access to all the information, or to none of it. With separate tables you can give certain people access to certain tables. This increases the security of the data. When looking at classes in the schedule of classes at Wayne State you only get to see information from the course table. You do not know the teachers name or office.

Customer-table	SS#	Name	City	Phone-number

Movie-table	Title	Year	Length	Rating	Director

Stock-table	Title	Format	Number

Rental-table	SS#	Title	Format

Figure 8.4: Tables for Video Store

8.3 Another Example

Here is another situation where a database would come in handy. What if you were to open up a video rental store. You would certainly need a computer to keep track of all the customers, and videos. Before you read on, take a couple of minutes and think about the kind of information that you will need to store.

We are certainly going to need a relation that stores information about each customer. We will also need a relation to store information about each movie we have in stock. We will need a third relation to keep track of how many tapes we have in each format and a fourth to tell us who rented what. That's right ...four tables. Figure 8.4 shows what we get.

The customer table holds all the information about a particular customer. the movie table holds all the information about a particular movie. The stock table tells us how many copies of each film we have in each format (LD, VHS, S-VHS, Beta, ED-Beta, Hi-8, etc.) The rental table tells us which customer rented which film in which format. Now you can see that the customer can be given access to the Movie and Stock table to help him make his selections. The Customer and Rental tables contain information that only the store owner should know. We could have entries like those in Figure 8.5.

Each customer is only listed once in the customer table. Each movie is only listed once in the movie table. The stock table has one tuple for each Format of each title in stock. The Rental table has one tuple for each video that has been rented. Each table is devoted to holding a specific type of information. We have security, we have convenience, and we have easy update.

8.4 Query Language

Given that we have a database set up, we need some way to access and modify the information that it contains. To do this we use a query language. It is the language

Customer-table

SS#	Name	City	Phone-number
453-87-6553	Hamner	Detroit	332-4378
346-84-5456	Jellison	Clinton	548-6562
235-63-7526	Czescu	Ann Arbor	878-8230
548-64-2315	Baker	Hamtramck	843-4105

Movie-table

Title	Year	Length	Rating	Director
Maltese Falcon	1941	102	n/a	Huston
Yellow Submarine	1968	87	G	Dunning
This is Spinal Tap	1984	82	R	Reiner
Die Hard	1988	127	R	McTiernan

Stock-table

Title	Format	Number
Die Hard	Beta	10
Die Hard	VHS	12
Die Hard	LD	2
Die Hard	S-VHS	5
Maltese Falcon	VHS	2
Maltese Falcon	LD	1
This is Spinal Tap	LD	1
This is Spinal Tap	Hi-8	2
Yellow Submarine	VHS	1

Rental-table

SS#	Title	Format
235-63-7526	Die Hard	LD
235-63-7526	Maltese Falcon	LD
548-64-2315	This is Spinal Tap	Hi-8
453-87-6553	This is Spinal Tap	Hi-8

Figure 8.5: Database for Video Store

in which we express our commands to the database. While the languages look fairly English-like, they are very strict. One of the current popular query languages is SQL ("sequel.") Let us say we were using the SQL language to talk to our video database.

Say we wanted to know the names of all the movies we have:

```
SELECT Title
FROM Movie-table
```

Say we wanted to know the names of all the R rated movies we have:

```
SELECT Title
FROM Movie-table
WHERE Rating = "R"
```

The query language allows us to communicate our request to the computer, but we must follow the format that the computer expects. Query languages have a very specific syntax that must be followed, so there is no ambiguity in your request.

8.5 Key

What is a key? A key is a column or set of columns that allows us to uniquely identify each row in a table. When the IRS processes your report, they are not going to use your name. With 250 million people in the US, it is likely that your name is not unique. Instead they will use your social security number. Your social security number uniquely identifies you. Keys are important in databases since they allow the computer to search through a table very quickly. For keeping your student records, Wayne State uses your social security number too.

8.6 Databases and You

As we find we need to manage more and more information, we will have a greater need for database programs. Currently the average person does not need a database. Large business have used databases for years, and smaller businesses are also seeing their benefits. As databases become simpler to use, and the hardware necessary to

run them becomes less expensive more and more people will find a use for them.

8.7 Questions

1. Give two situations where it would be good to use a database, and why?
2. What are databases best used for?
   (a) drawing pictures with awesome graphics
   (b) efficiently storing and retrieving large amounts of information
   (c) storing small amounts of information for quick access
   (d) writing term papers with lots of tables
3. Give three reasons why having many small tables is better than one large one?
4. What is a query language?
5. What does a key of a table allow us to do?
   (a) uniquely identify a column
   (b) uniquely identify a database
   (c) uniquely identify a field
   (d) uniquely identify a row

Chapter 9

Spreadsheets

While you will probably find that word processors are the most useful kind of application, spreadsheets can also be very useful. Spreadsheets allow you to do work with numbers in a very comfortable way. They allow you to do complex calculations simply, and quickly (after all, everyone knows computers are good at math.)

The first spreadsheet program, called Visicalc, was written by Dan Bricklin and Robert Frankston. Visicalc was short for VISIble CALCulator. It was released for the Apple][in May 1979. This program was a top selling program for years, and was the biggest reason that businesses began to take a serious look at personal computers. The current standard in spreadsheets, called Lotus 1-2-3 was released three years later. Within this chapter we will be using the conventions of the Microsoft Excel spreadsheet which is probably the best spreadsheet currently on the market. Visicalc was so well thought out that all of its successors, including Lotus, and Excel, have almost identical commands.

9.1 Terminology

A spreadsheet looks like an accountant's ledger pad. It is made up of a grid of cells. Each **cell** can be uniquely identified by its **row** and **column**, and each cell contains either **text**, **numbers** or a **formula**. Numbers and text we have seen before. A formula is used to make calculations.

	A	B	C	D	E
1	Student's Name	Exam #1	Exam #2	Exam #3	Average
2					
3	Jervas Dudley	78	76	86	
4	Basil Elton	56	49	65	
5	Karl Heinrich	81	66	84	
6	Thomas Malone	89	87	86	
7	Lavinia Whateley	53	86	75	
8	Asenath Waite	99	89	95	
9					

Figure 9.1: Student Grade Spreadsheet

9.2 Example

Let us say we wanted to set up a spreadsheet to figure out the grades for a class. We assume that each student has three exam grades and we need to find out what the average score is. We could set up data in a spreadsheet as shown in Figure 9.1.

As you can see, each cell has a specific address given by its row and column. Cell A1 in Figure 9.1 has the text "Student's Name" stored in it. Cell D8 has the number 95 stored in it.

Typing information into a cell is simple. You select the cell you want (using either the arrow keys or a mouse) and then type what you want the contents to be using the keyboard. When you are done typing you hit the enter or return key. We can set up the simple table shown in Figure 9.1 this way.

Now we need to find the average score for each student. For Jervas we need to take 78 + 76 + 86 and then divide by 3. We could do this on a calculator and then type the number 80 into cell E3.

But we can do it easier than that. We could type a formula into cell E3 that says "=(78+76+86) / 3" and it would calculate the average for us. Then we could do the same for the other five students.

Well, we can do it even easier than that. Instead of typing numbers into the formula we can type the cell locations into the formula. We could type a formula into cell E3 that says "=(B3 + C3 + D3) / 3" and it will calculate the average for us. Cell B3 contains the number 78. Cell C3 contains the number 76. Cell D3 contains the number 86. This way has a big advantage over the other two methods. What happens if Jervas' score on Exam #3 is wrong and he actually deserves 90 points. We need to change the value in cell D3 to 90. That's simple. We select cell D3 and type 90. What about cell E3 - do we need to change the formula? Not if the formula

is "=(B3 + C3 + D3) / 3" because this formula has no numbers in it ...only cell locations. Since the value in location D3 has changed, the spreadsheet automatically recalculates all the formulas. If we used the other methods we would need to retype the formula in cell E3.

Now you may be wondering what appears in cell E3. We have typed in a formula, but we expect it to calculate a number for us. The number is what appears in the cell, but when you select the cell you have the opportunity to modify the formula. You can think of the formula as a third dimension to the two-dimensional spreadsheet.

This is the big advantage spreadsheets have over calculators and ledger pads. You can very easily vary the numbers and see how the results of the formulas change. You can find out what would happen if this or that happened. So, what would the formula be that we type into cell E4 to calculate the average of Basil's exam scores? I hope you guessed:

=(B4 + C4 + D4) /3

That's the correct answer. We can write similar formulas for the other four students.

You might have noticed that the formulas all have "=" signs out in front. That is how the spreadsheet differentiates a formula from text. All formulas must start with an "=" sign.

9.3 Circular Reference

What if we have the following situation:
Cell A1 contains "=B1 + 2"
Cell B1 contains "= A1 - 1"

The computer can not calculate the value for cell A1 without the value for cell B1. It can not calculate the value for cell B1 without the value for cell A1. This is called a circular reference. It is bad. It will cause the computer to give you an error message because it can not complete its calculations. This brings up the question of how does the spreadsheet know what to calculate first. The spreadsheet looks through all the calculations that it must do and does all the ones it can do, then it looks back at the calculations it couldn't do before to see if there are any that it can do now, having done the previous batch. It keeps on going until it has done all the calculations.

9.4 Relative Referencing

Now we have to talk about absolute referencing and relative referencing. These are the two ways to refer to a cell in the spreadsheet. What we have shown so far is relative referencing. Let us look at the first formula we wrote. We put a formula in cell E3 that said:

=(B3 + C3 + D3) / 3

Now the spreadsheet interprets this formula to mean:

take the value in the cell three to my left and zero above me
and
add it to the cell that is two to my left and zero above me
and
add it to the cell that is one to my left and zero above me
and
divide the total by three.

The locations of the other cells (cell B3, cell C3 and cell D3) are shown relative to the cell the formula is written in (cell E3). Now you may think that this seems needlessly complicated. The advantage comes with the ability to copy and paste formulas from one cell to another. What happens if we copy the contents of cell E3 into cell E4. Well now cell E4 contains:

take the value in the cell three to my left and zero above me
and
add it to the cell that is two to my left and zero above me
and
add it to the cell that is one to my left and zero above me
and
divide the total by three.

That gives the correct answer for the second student without us having to retype anything. If we click on cell E4 to look at the formula we see that it is

=(B4 + C4 + D4) / 3

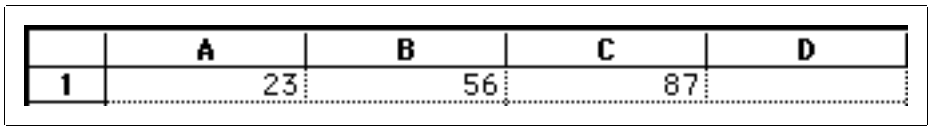


Figure 9.2: Standard Deviation Needed

So for our grades table all we have to do is type in one formula and copy it into the other five cells, and the formulas will automatically change to fit their new location.

9.5 Absolute Referencing

But what happens if we do not want a formula to change when we move it? What if we do not want to refer to the cell three to my left and zero above me, but we really want to refer specifically to cell A3? In this case we would write the row and column with "$" signs. For example the formula in cell E3 could have been written:

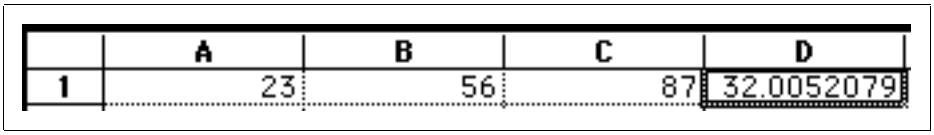
=(B3 + C3 + D3) / 3

and it would have given the correct answer. But what if we copied that formula into cell E4. The formula in cell E4 becomes:

=(B3 + C3 + D3) / 3

This is the same as the formula in cell E3. We are referring to the row and column absolutely so there is no automatic changing of the formula.

Spreadsheets can go beyond simple arithmetic because they have built in functions. This makes them very useful for college students with lab courses. In Physics labs there is very often a need to take the standard deviation of a set of numbers. Now the standard deviation is a nasty formula which I will not reproduce here for fear that some students will suffer an immediate panic attack. With a spreadsheet you don't have to worry about how to calculate the standard deviation ... let the computer do it for you. Lets say we have the situation shown in Figure 9.2. and we want to have the standard deviation of the three numbers in cell D1. This is really simple. We type "=STDEV(A1, B1, C1)" into cell D1 and we get our answer as shown in Figure 9.3.



	A	B	C	D
1	23	56	87	32.0052079

Figure 9.3: Standard Deviation Obtained

average	sin
absolute value	cos
exponents	tan
logarithms	
maximum	standard deviation
minimum	variance

Figure 9.4: Spreadsheet Functions

STDEV is the name of the function that calculates standard deviations. There is another function we could have used back with the grades example. In cell E3 we could have typed:

=AVERAGE(B3, C3, D3)

instead of

=(B3 + C3 + D3) / 3

and we would have gotten the same answer. Spreadsheets have many built in functions to make calculating easier. Some of these functions are shown in Figure 9.4.

9.6 Spreadsheets and You

So what can you really do with a spreadsheet?

- Balance your monthly finances

- Calculate data for your lab report
- Figure out your taxes

Spreadsheets conveniently organize numerical information and allow us to "play" with it. They allow us to ask questions like "What happens if ..." and get immediate feedback. Anyone who deals with numbers on a regular basis should think about investing the time to learn how to use a spreadsheet program. The time you save, will be your own.

9.7 Questions

1. What is the difference between relative referencing and absolute referencing?
2. What are the three different things that a cell can contain?
3. Which of the following does NOT calculate the average of A1, B1 and C1
   (a) =A1 + B1 + C1 / 3
   (b) =AVERAGE (A1, B1, C1)
   (c) =SUM (A1, B1, C1) / 3
   (d) =SUM (A1:C1) / 3
4. What is the difference between the name of a cell and its value?
5. A circular reference
   (a) is used to calculate the area of circles
   (b) is used to calculate the radius of circles
   (c) is used to calculate the geodetic prime arc of a circle
   (d) will cause a spreadsheet to print an error message
6. Which of the following types of referencing should be avoided?
   (a) Absolute
   (b) Relative
   (c) Circular
   (d) Absolute and Relative

Chapter 10

Graphics

In the 60's Ivan Sutherland created Sketchpad - the ancestor of all our modern graphics programs. When personal computers came along, the ability to draw shapes with pretty colours on the monitor screen became a great selling point. Unfortunately the early programs were very primitive, and the early monitors did not have the resolution to produce anything beyond blocky drawings in a handful of colours.

10.1 Painting Programs

The first really useful, and popular, painting program for personal computers was MacPaint, released with the Macintosh in January 1984. This program was the easiest computer program to use at the time, and was capable of producing some very nice artwork. This program was soon imitated, and today there are several similar painting programs for every personal computer. The problem with MacPaint (and the early Macintoshes) was the lack of colour. You could only paint with black and white patterns. This left the door open for a big improvement which came one year later with the release of the Amiga. The Amiga ran colour painting programs, with many sophisticated painting options.

Paint programs work by allowing you to set the colour of each pixel on the screen. They also allow you to draw lines and curves, and fill areas with a certain colour. They give you several shapes and sizes of "brushes." They let you move and rotate parts of your picture. These pictures can then be printed out or, more importantly, copied into other documents such as a report you are typing with a word processor.

Figure 10.1 shows the set of tools you get with MacPaint, along with some of the

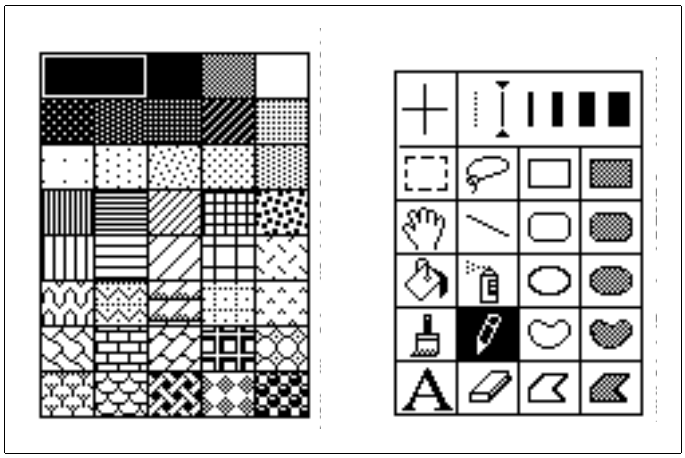


Figure 10.1: Paint Patterns and Tools

sample black and white patterns. When we create artwork in the real world we have several tools available to us: paint brush, pencil, eraser etc. On the computer we usually have a mouse. By selecting a tool you "transform" the mouse into a pencil or a paint brush or an eraser, depending on the work you need to do.

10.2 Digitizing

Now having this painting program is very nice, but no matter how friendly the program is, you still need artistic talent. But what about those of us who have very little artistic talent? A digitizer allows us to take a picture and convert it into a paint file. The analogue picture (say a photograph, or a picture out of a newspaper) is converted into a digital one for the computer. The picture will be a bit grainier than the original, but now it can be modified using all of the tools the paint program gives us. The higher the resolution of the screen and the more colours that are available - the closer the digitized image will be to the original. Since the picture is now in a digital form, we can move it into word processing documents, or upload it to a BBS. Figure 10.2 shows an example of a digitized image. Figure 10.3 shows that same image after it has been modified using a paint program.

Officially, digitizing is the conversion of anything to a digital form. Most commonly this is a picture, but it can also be sound. Most personal computers can play digitized sounds, as well as display digitized pictures.

Figure 10.2: Digitized Image

Figure 10.3: Modified Digitized Image

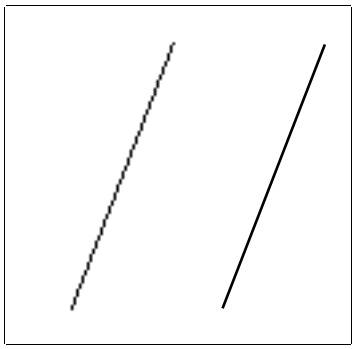


Figure 10.4: Painted Line VS Drawn Line

10.3 Drawing Programs

There is another type of graphical program called drawing programs. Drawing programs became very popular with the arrival of Laser Printers, and the first popular one was called MacDraw. Along with Word Processors and Laser Printers, Drawing programs are the three integral parts of Desktop Publishing. Unlike Paint programs, drawing programs do not alter individual pixels. Instead drawings are held in terms of their mathematics. This allows the drawing to have as much resolution as the printer can supply. Where painting programs work well on a monitor with its 72 dots per inch resolution, these images do not look very good on a printer with 300 dots per inch resolution. Paint files tend to look grainy when printed on Laser Printers. Draw files look very fine. The difference is shown in Figure 10.4. The line on the left was created with a paint program. The line on the right was created with a draw program. On the computer screen they look identical, but when printed onto paper there is a big difference.

Back in school you did many different things in art class. You could make a picture with a paint brush and several bottles of paint. You could also make a picture by cutting shapes out of coloured construction paper and then arranging these pieces of paper. This is the difference between a paint program and a drawing program. The paint programs puts colours onto a canvas. The drawing program moves objects around on the canvas. With a drawing program you can draw a box and a triangle. You can then grab the box and move the box under the triangle to create a house - just like you did back in art class. If you don't like the way it looks the drawing program lets you stretch the box, or move it somewhere else. Like using construction paper you can put one object on top of another, and move them around till they

look the way you want them to. The advantage of a drawing program over using construction paper is that you never have to glue the paper down - you can always come back later and move the pieces around.

Artistic drawings tend to be done with paint programs since paint programs give more kinds of tools for putting colour onto the screen. Technical drawings tend to be done with drawing programs since they give more kinds of tools for moving and modifying objects.

10.4 Desktop Video

Now the usefulness of painting programs extends beyond printing out diagrams. Paper is not enough to keep people's attention today. People want colour and movement and sound. They want animation. The personal computer can be an excellent tool for creating exciting presentations. As the Macintosh made "desktop publishing" possible, the Amiga made "desktop video" possible. Computer animation is done in the same way that animation has always been done, except that all of the cells are created with a computer and stored on disc. Rapidly showing these cells in order on the monitor screen gives the illusion of movement. The computer can then be connected to a VCR to record the animation.

You have probably all seen the computer generated intros that all the major television networks have. You might have even seen theatrical films such as "Tron" or "the Last Starfighter" which both made extensive use of computer graphics. Big computers can truly produce some "awesome graphics" (as a former CSC 101 student put it.) You may be surprised to learn that some personal computers can produce some pretty awesome graphics of their own. Local TV stations have found personal computers (especially the Amiga) to be very useful in generating professional looking animation at a fraction of the cost for the professional studio.

10.5 Rendering

Rendering is a general term used to describe the process of creating a graphical image on the computer screen. Artists consider using a paint program to be rendering where computer scientists use the word to describe the process of generating images by some computational method such as ray tracing.

Ray tracing is the process of realistically (and mathematically) rendering an image

using a geometric 3-D model. Typically a 3-D model is created out of polygons which are "glued together" to form "solid" objects. The ray tracer then "fires" rays of light into the scene and traces the path each ray takes as it reflects off or refracts through the various objects. This way the computer can create much more realistic looking scenes than a human artist can. Ray tracing is particularly good at rendering glass and metallic objects. Combining these rendered images with animation can create some very impressive presentations. Figure 10.5 shows a computer generated image of a desktop complete with desklamp, books, and bouncing-silver-ball-thing. Note the reflections in the silver balls. The actual image is in colour and has been reduced to black and white for inclusion in this text.

Rudimentary rendering is possible on IBM-PCs with more advanced rendering possible through TrueVision's TARGA boards. High quality rendering is possible on the Mac II, but at a somewhat inflated prices. High quality rendering is also possible on the Amiga at more reasonable prices. Production quality animation is done on $20,000 systems such as Silicon Graphics' Iris.

10.6 The Making of George

The following is a description of how a cartoon character is designed and generated using the computer.

First of all a special program called a **modeller** is needed. A modeller is a program that allows the user to create three-dimensional objects that resemble objects in real life. Modellers are commonly used today to design cars because it allows the user to visualize the finished product before it is built.

In our example we have built a cartoon character called George who is nothing more than a blue ball with bulging eyes. Using the modeller, we first build George's body. Modellers typically have a built in library of objects that can be loaded at any time. Since George is shaped like a ball, the modeller can simulate that shape using a sphere as shown in Figure 10.6.

Notice that the sphere is not perfectly smooth, but instead is composed of facets. These facets are called **polygons**. All three-dimensional computer generated objects are created using polygons. When there are enough polygons strung together the final object will look smooth, like the sphere.

Next we will create George's eye balls. George is actually a very simple cartoon character to build with a modeller. His eye balls are simply two more spheres which will be attached to his body (the larger sphere we created earlier) at the appropriate

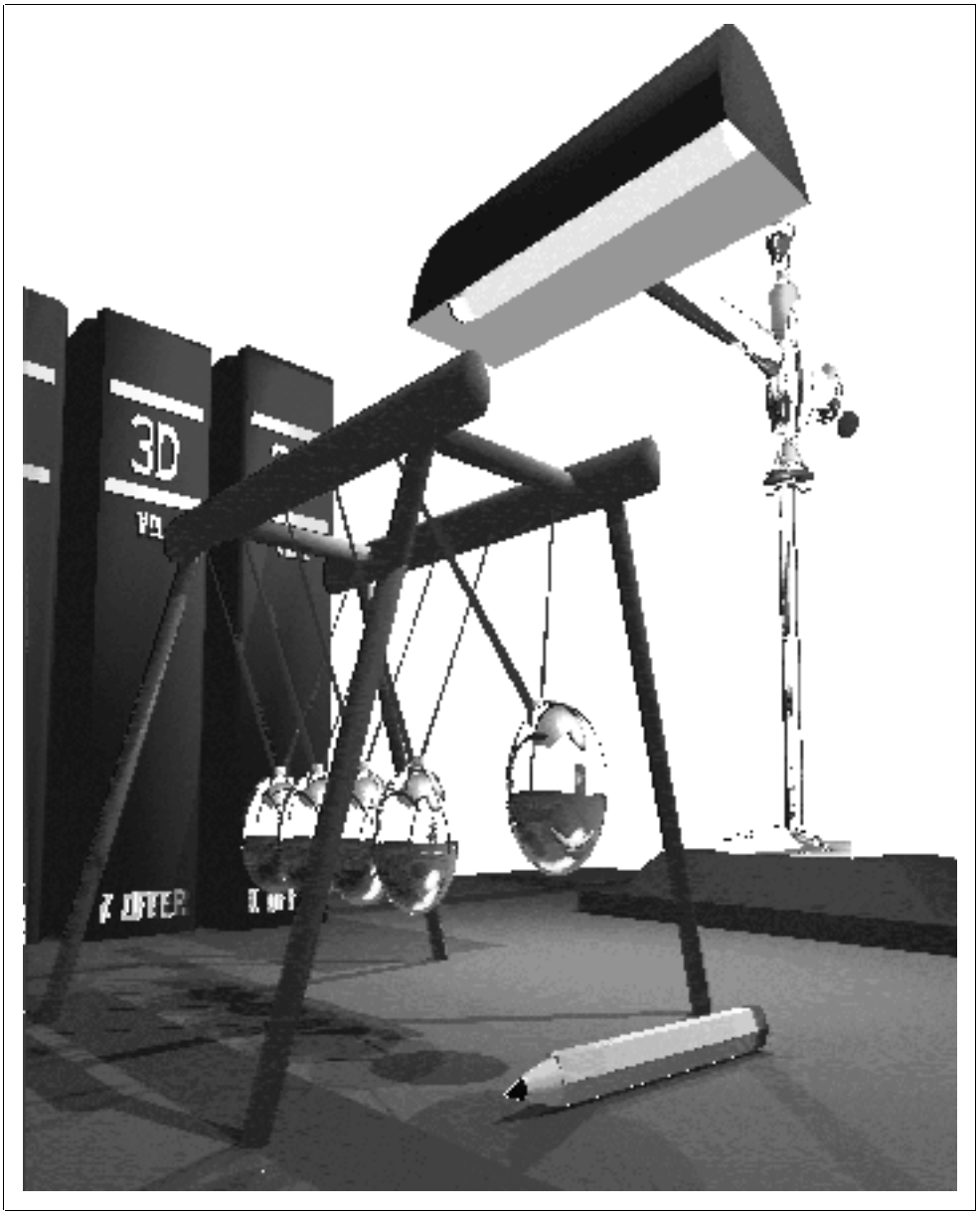


Figure 10.5: Ray Traced Desktop

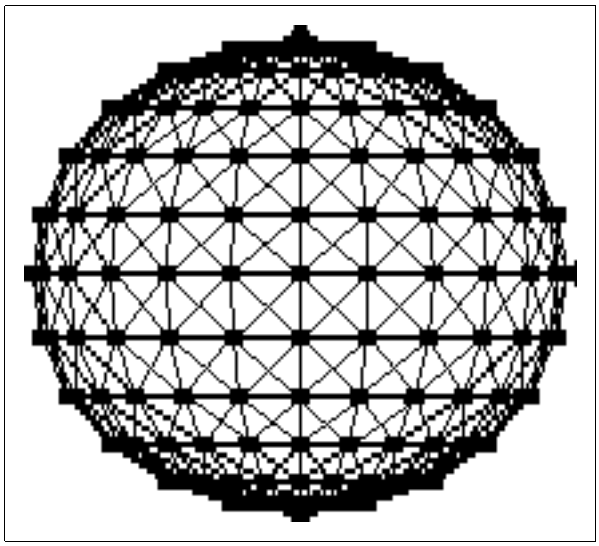


Figure 10.6: Simple George

points. Figure 10.7 shows this new version of George.

Now would be a good time to have a look at how George looks before we put any more work into him. First of all we need to tell the modeller where we would like to put our camera (to "take the photo") and where we would put the lamps to light up the scene. The camera is placed directly in front of George and the light source is placed to the left and above the camera. This is shown in Figure 10.8. Having done this we can get a quick preview of how George looks in 3D. This is done using a **wireframe model**. A wireframe model is like the skeleton of a large ocean liner before sheets of metal are bolted to its sides. The wireframe version of George is shown in Figure 10.9.

So far George looks pretty good so we can proceed by putting George's pupils in. These are nothing more than two disks; one for each eye as shown in Figure 10.10.

We are almost ready to take George's photo (which is called rendering in computer lingo); but before we can do that we must assign colors to George. George's body will be blue, his eye balls will be white and his pupils will be black. These kinds of attributes are assigned using a control panel as shown in Figure 10.11. In this case the panel is showing the settings for George's body which is a smooth glossy blue ball.

Finally we are ready to render George. We are going to render him using a technique called **ray tracing**, which is the most time consuming process in computer

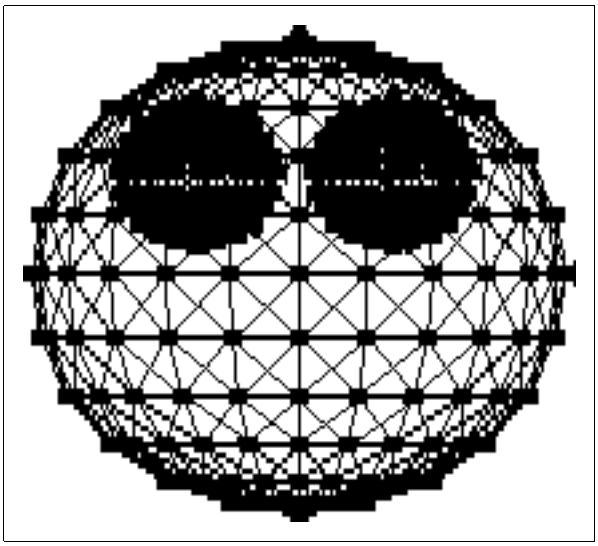


Figure 10.7: George with Eyeballs

graphics. Where a wire frame can be generated in seconds, a ray traced image can take days to generate. This is because in order to simulate what you see in the real world the computer has to simulate each point of light that leaves your TV screen and hits the three-dimensional object that you have designed. If the object itself is reflective then the ray tracer must continue by further tracing the path of the reflected light. A lot of Physics, Mathematics and Fudging is involved in successfully ray tracing a truely photo-realistic scene. Because George is a relatively simple object, he only takes 1 minute to ray trace on an Amiga with a Motorolla 68030 CPU and 68882 Floating Point processor. Our final version of George (reduced to black and white for this text) is shown in Figure 10.12. On the computer screen George is a shiny blue ball with bulging white eyes.

In order to create animation, the camera and/or objects must be carefully moved in the three-dimensional world. After each movement has been established, the ray tracer can then render each individual frame of the image. If a frame takes 24 hours to render, it would mean that it would take about 30 days to simply generate 1 seconds worth of animation (assuming 30 frames a second - full movie speed). And a full length movie that lasts about 2 hours would take about 216000 days to create ;which is about 592 years! That is why for the most part you will see computer graphics and animation as something that enhances a live-action movie (like The Abyss) rather than becoming the entire movie. The computers we use today for computer graphics

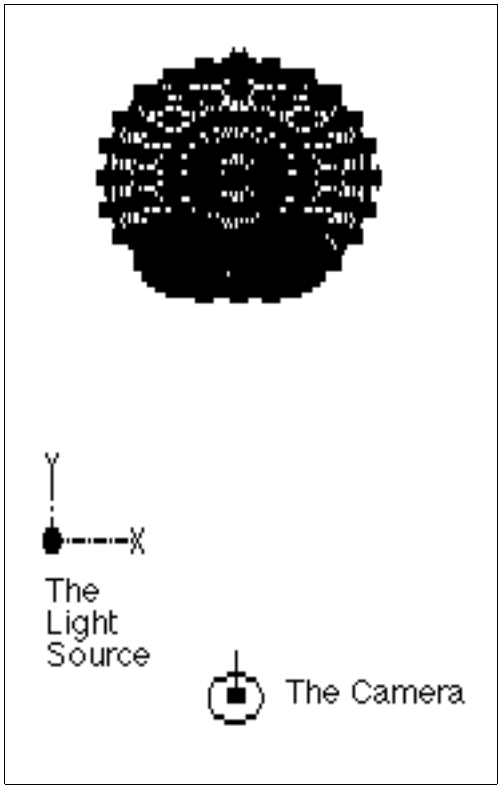


Figure 10.8: Top View Showing the Positions of the Light and Camera

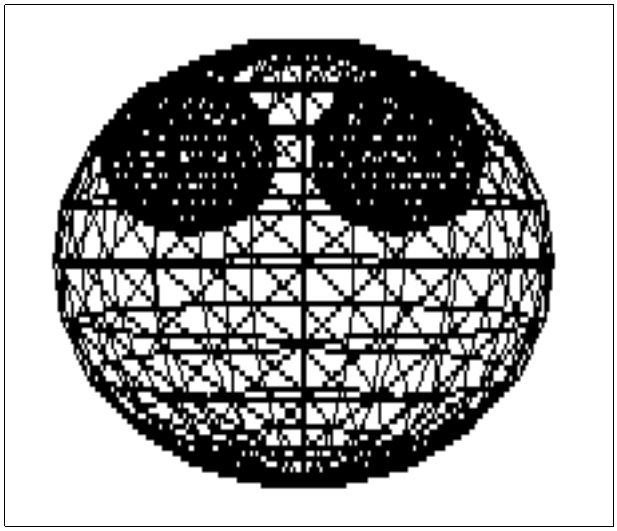


Figure 10.9: Wireframe Version of George

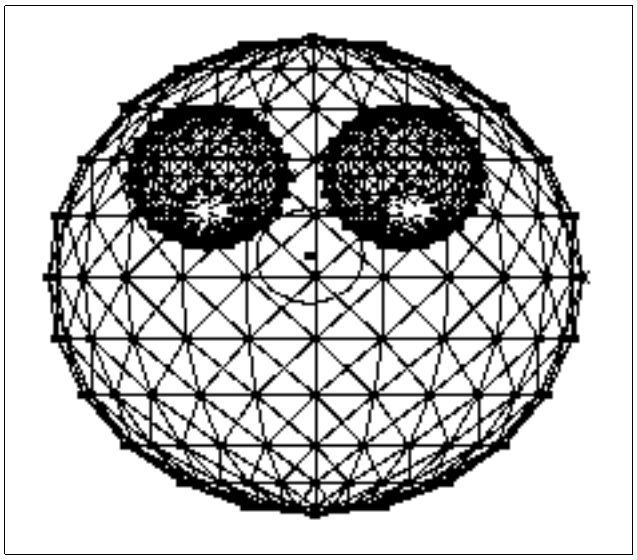


Figure 10.10: George with Pupils

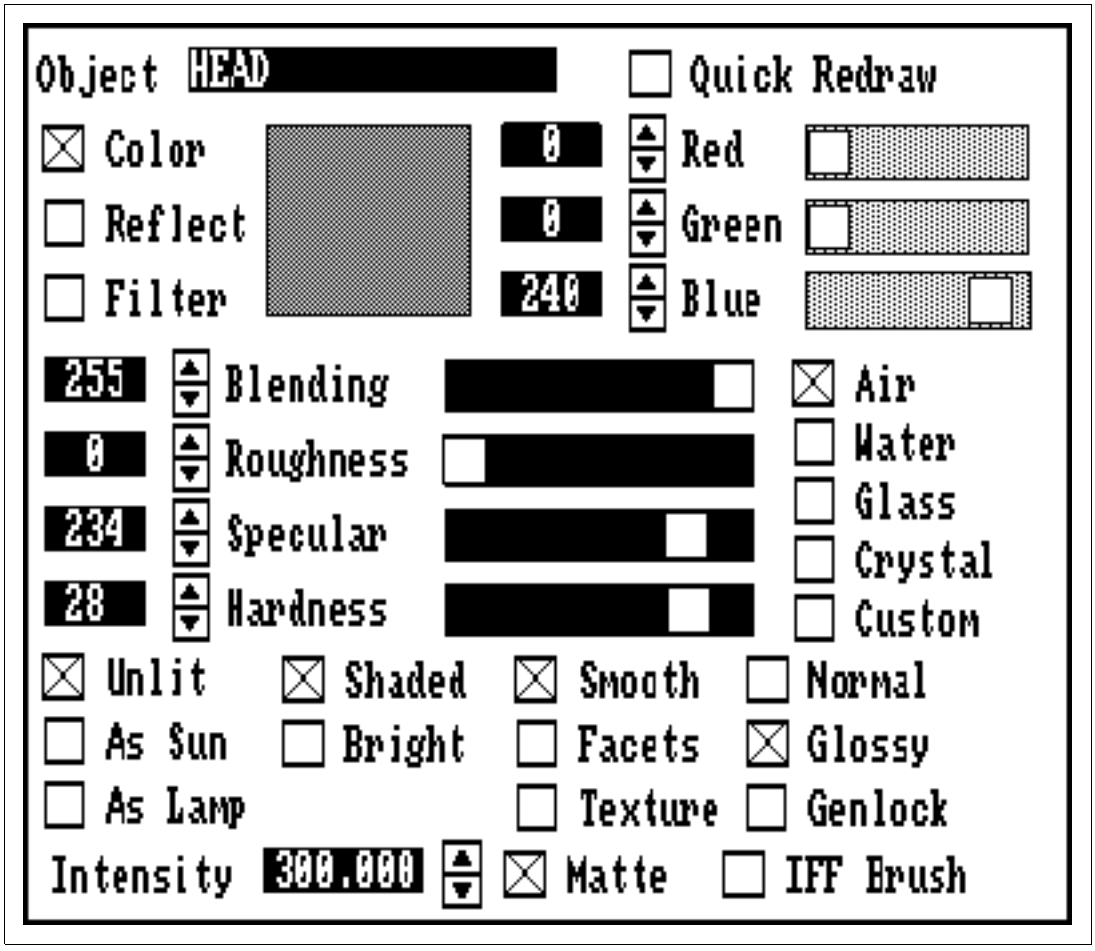


Figure 10.11: Setting up George's Attributes

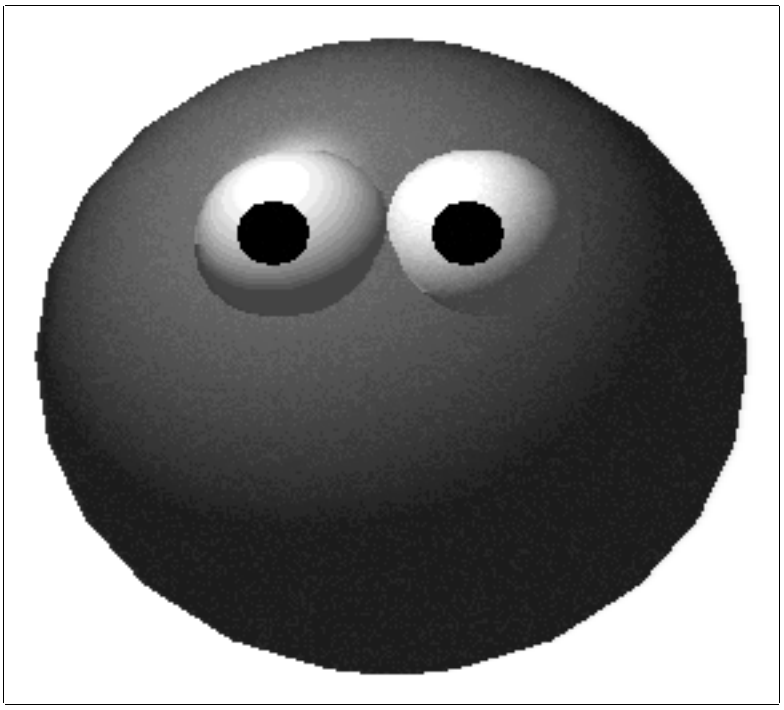


Figure 10.12: Ray Traced Version of George

are very fast; but still not fast enough...

10.7 VideoGames

Now we certainly couldn't have a section on graphics and not talk about videogames. The current generation of home videogame machines have the same CPUs as personal computers do. In fact, all they need is a keyboard and a disc drive to really be considered a personal computer that specializes in graphics and sound.

Videogames require a lot of computing power, as they typically involve fast multi-colour animation along with multi-voice stereo sound. Remember that back in the mid 70's Pong was cutting-edge videogame technology. For those of you that missed the 70's, a typical "table tennis" video game is shown in Figure 10.13. Today one of every four homes in America has a videogame system. This is where microcomputers have really found a home ...in your home.

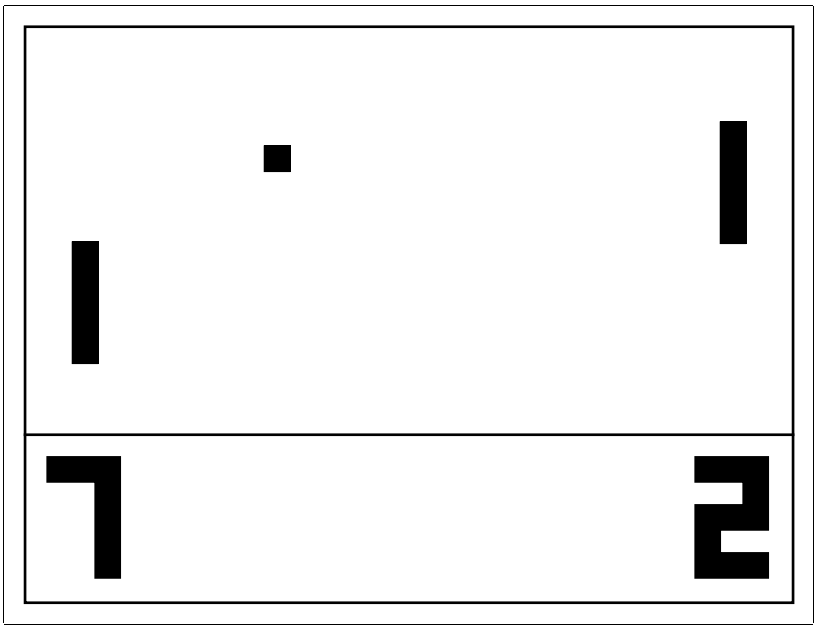


Figure 10.13: Ye Olde Video Game

10.8 Graphics and You

Good graphics require a lot of computing power, and as machines become faster, graphics quality will continue to improve. As desktop publishing brought the power of a small printshop to your desk, desktop video now brings the power of a small video production facility. The creativity and talent still rest with the individual artist. The computer simply gives you another medium to work in and another set of tools to help you express yourself.

10.9 Questions

1. What is rendering?

2. How does a drawing program differ from a painting program?

3. What is digitizing?

   (a) converting audio or video into a form the computer can understand

(b) multiplication using binary numbers

(c) the act of typing numbers into the computer

(d) the method Ted Turner uses to 'improve' old movies

Chapter 11

Hypertext

We have said that computers are good at managing information, and there is a lot of information out there to manage. We can use a computer to help us get to important information quickly. A book is divided into chapters with a table of contents at the front. Why? It lets us find the information we are searching for quickly. Textbooks have indices in the back. Why? It lets us find the information we are searching for quickly. It would be very time consuming to have to start at the beginning of a book and keep reading until we found what we are looking for. Tables of Contents and Indices allow us to take shortcuts, and go directly to the piece of information we want to look at. It would be nice if we could take even more shortcuts.

Even in 1945 the need for a system that cross referenced information was apparent. Vannevar Bush came up with the idea for a system called Memex. This system would allow the linking of pictures and text. Unfortunately the technology of 1945 would not allow this system to be built. The first working hypertext system was created in 1967 at Brown University.

11.1 Hypertext

In many papers we make use of footnotes. We put a number into our text and the reader can use that number to find the corresponding footnote reference. The footnote number acts like a link between the main body of the text and the additional information stored in the footnote itself. Encyclopedias work on this same principle. They list related topics that you can look at for further information. The limitation of printed material is that you must go to the related piece of information. Hypertext

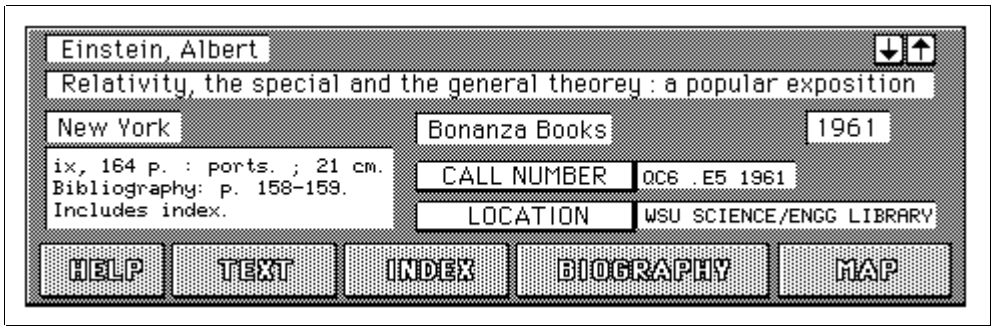


Figure 11.1: Card Catalogue

is a generalization of these concepts.

Ted Nelson coined the term "Hypertext Systems" and it was appropriate for the early systems that were developed since they could only link text. Officially hypertext is non-sequentially linked pieces of text or other information. Hypertext systems consist of information and links that link related pieces of information together. A person does not have to start at the beginning and read through to the end. He can start at the beginning and then go in the direction that interests him. Using a hypertext system is often known as "navigating through an information space." Information is linked together, and the links allow the user to move through the sea of information, only looking at those pieces of information that are of interest, and moving quickly between related interesting topics.

Hypertext systems display their information on the monitor screen so the user does not have to go searching for the related information. The computer does the searching and displays the requested information. Most of the monitor screen for a hypertext system is devoted to displaying the requested information, and the rest contains buttons allowing the user to choose what to look at next. Since it is easy to get lost in a large hypertext document maps are often provided on the screen to show you where you are and where you can go from here.

For example, we could have a hypertext card catalogue for a library. The information about each book is displayed on the monitor screen in the form of a card as shown in Figure 11.1.

As well as having the standard card catalogue information which would allow the user to search for books based on various criteria, this system may have links allowing you to read an abstract from the text stored electronically. At the push of a button you would have the opening chapter of the book displayed on the screen in front of

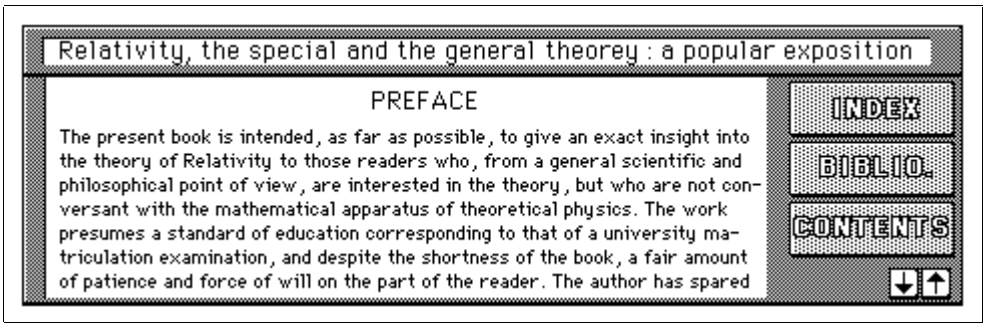


Figure 11.2: Text of the Book

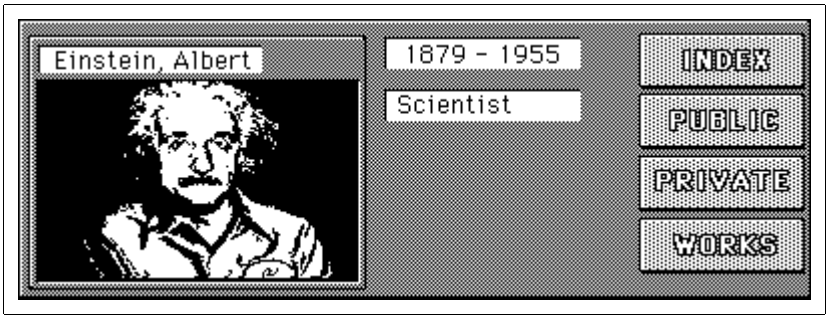


Figure 11.3: The Author

you as shown in Figure 11.2.

The names of important people and events in this book would be linked to information about them. This is where the original concept of hypertext comes in. Certain important words have links to additional information, or other sections of the text.

Another button could bring up biographical information about the author as shown in Figure 11.3.

Of course, a system like this is still a few years off, but primitive versions are currently in operation.

Each user charts her own course through the information. We said before how modern computers allow us to work in ways that are intuitive to us. Hypertext systems allow us to search for information in a more natural way. Hypertext systems also put a "friendly" face on the monstrous amount of information that is accessible. With computers becoming faster, storage devices becoming cheaper, and interfaces being easier to use, we can look forward to seeing more and more of these hypertext systems becoming available.

One the new "buzzwords" of the 90's is **multimedia**. Why not allow the computer access to information from different media. For example connect the computer to a compact disc player, or a laser disc player. Let's use the computer to manage other types of information aside from plain text. Multimedia applications are supposed to be the hot computer area of the 90's and Commodore is already pushing their newest Amiga as a "Multimedia platform." Computers will be managing larger, and more varied amounts of information in the coming years. The more information available, the more useful a hypertext system will be to help you search through it.

11.2 Hypermedia

Apple's Hypercard was the first really popular hypertext program and the first program to successfully expand the idea of Hypertext into Hypermedia. The information that a hypertext system moves through does not have to be stored within the computer. Hypermedia is a combination of hypertext systems to a multimedia computer. Hypermedia allows the user to easily access information from many different forms of media. Hypercard allowed computer users to run Compact Disc players and their larger siblings Laser Disc Players from a Macintosh. A laser disc can hold 50,000 full colour still pictures on a single side. Trying to look up a certain frame on a printed index would be very difficult by hand, but a hypermedia system would make it simple.

The Voyager Company has had some HyperCard hypermedia products on the market since 1988. One of their discs contains everything you ever wanted to know (and much, much more) about Beethoven's 9th Symphony. Other currently available products include a complete dictionary on CD-ROM. Not only does this dictionary give you the standard definitions and origins for a word, it also gives you an audible pronunciation. You don't have to translate the pronunciation symbols anymore because the dictionary will "speak" the word for you. An entire encyclopedia can be stored on disc along with full colour pictures. Since the computer can access this information for you, you can search much more quickly. And they have the advantage of being portable. Why lug around a 2000 page dictionary or a 15 volume encyclopedia when you can carry around a compact disc?

11.3 Memex

It seems hard to picture how a hypertext system could have been thought up in the 40's by Vannevar Bush. Computers were very rare and very expensive things and had nowhere near the capabilities of today's desktop personal computers. Vannevar Bush's system was designed based on the technology of the times. Memex was a desk that the user would sit at. Information in the form of pages would be stored on microfilm. Using a lever you could show one of these pages on one of the translucent screens of the memex. The machine you use at the library to look at microfilm works just this way. But how do you get information into the machine? Through photography. The memex would be able to take a photograph of any piece of paper and store it on a new piece of microfilm. You could bring up a page onto one of the screens, add some new notations, and then take a new picture of the changed page.

The hypertext links were kept through coded numbers. A book on the desk would contain the numbers of different pieces of microfilm. Each piece of microfilm would have its own numebr as well as the numbers of related pieces of microfilm. If you were interested in 'following the links' you would then look up that piece of microfilm.

I find the ideas behind the memex interesting because it would have been an information processor that didn't not rely on digital processing or even anything remotely similar to a computer of today. Today however you could make a much more powerful memex system. Microfilm would be replaced by hard discs and CD-ROM. Photography would be replaced by digiters. Output could be onto full colour monitor screens or hardcopy onto a laser printer. The built in computer could access and display this information quickly. Changes could be made to the documents using the keboard, an optical pen or a mouse. And, of course, the 'desk' could be compacted down to the size of a briefcase for portability.

11.4 Hypertext and You

Hypertext products also offer the promise of allowing people to create multimedia presentations with very little programming knowledge. Hypertext systems come with their own programming languages to allow their users to create new hypertext documents. This is good for non computer science people who want to create hypertext information systems. Hypertext systems have found their first niche in museums, where the museum can set up hypertext systems for their patrons to use. Each visitor to a specific exhibit can use the system to look for more information in specific

areas she is interested in.

Unfortunately all the hype surrounding hypertext and multimedia obscures the fact that there really is a need for systems such as these. Their usefulness in teaching could be astounding, but like every other tool they must be used appropriately. Hypertext and multimedia systems will give people easy access to vast amounts of information, but correlating these facts and learning from them will still be up to the individual.

11.5 Questions

1. What is hypertext?

   (a) an automatic linking of related pieces of information

   (b) the way text wraps around the screen when you use a word processor

   (c) sending information over phone lines

   (d) typing text into a paint program

Chapter 12

Programming

> Computers are like an Old Testament God; A lot of rules and no mercy.
> – Joseph Campbell.

One of the questions posed when the topic of programming crops-up is: Why do I need to learn how to program a computer when I can just buy any program that I need? Programming is much more than learning a set of "incantations" that tell the computer what to do. Being able to write a program, even a very simple one, allows you to take full control over the computer and make it do what you want it to do, rather than just making do with what others have already written. Programming will give you a better feeling for why programs work the way they do, and how much work goes into creating them. Programming also teaches the very important skill of breaking a problem down into subproblems - stepwise refinement. This skill comes in handy in all kinds of ways, as it forces you to think in a very logical and step by step manner. You will find that the more specialized your need, the less likely that a program exists to fulfill that need. Being able to write a program will allow you to create the program that you need, rather than waiting for it to be written by someone else. There is also a great market for programs written by people in specialized fields.

Programming is the art of writing computer programs. Programming is truly an art as well as a skill. When we write a program we are not only interested in solving the current problem. We want to write a program that is easy to read, easy to understand, and easy to modify if the need arises. We want an elegant solution to the problem. We are not just painting the garage to hide the cracks, we are painting a picture to hang on the dining room wall. We will not take the "Cliff Notes" approach to programming here.

12.1 Algorithm

Given a problem there are usually many methods that will solve it. Each person sees a problem in a different way, and will come up with a unique solution. When we are given a problem we try to decide on a sequence of steps that will solve it. In computer science this sequence of steps is called an algorithm. In the real world it is called a recipe. Computer scientists use the word algorithm because it makes them feel important. A program is the translation of this sequence of steps into a form that the computer can understand and perform.

Algorithms have existed for more than 2000 years. The first nontrivial algorithm was developed around 350BC by Euclid (remember him from Geometry?) when he invented a procedure to find the greatest common divisor of two positive numbers. The word 'algorithm' comes from the name of a 9th century Persian mathematician named Mohammed-al-Khowârizmî which was translated into Latin as 'algorismus.' Mohammed-al-Khowârizmî created step by step rules for doing basic math on decimal numbers.

Now, typically we aren't confronted by problems like those of Euclid or Mohammed-al-Khowâizmî but as human beings going about our business we often come up against new situations: How do I make chocolate chip cookies? How do I change the oil in my car? How do I register for classes?

12.2 Stepwise Refinement

The best way to solve these problems (and in fact the way humans generally DO solve problems) is to use a process called stepwise refinement, which is also known as "divide and conquer." We break down a large problem into smaller and simpler problems until we have solved all the small problems, and then we have (magically)

solved the larger problem. In effect we create a recipe, or algorithm, for solving this large problem - we find a sequence of steps that will solve it. We will be able to use this recipe again in the future if we come across a similar problem.

Imagine you have a friend who has never been to a movie theatre, and you want to tell her how to go about seeing a movie.

As in a recipe, there are certain 'ingredients':

- you need a movie that she wants to see
- she must be able to get to the theatre at the correct time
- she must have enough money
- she must be able to get home from the theatre

Assuming all these ingredients are available we can think about the steps involved. first we start off with something very general:

1)See a movie

Well, lets break (1) down into several sub-problems. First we need to go into the theatre, then buy a ticket, then go into the 'viewing room', then watch the movie, and finally we leave. Now we can write these down a little better as follows. What we are building up looks suspiciously like an outline.

refinement of see a movie
1.1)enter theatre
1.2)buy ticket
1.3)enter viewing room
1.4)watch movie
1.5)leave theatre

(1.1), (1.2), (1.3), (1.4), and (1.5) all refine step 1. We like to number the steps as in an outline so we don't get confused. We started with one problem, and now we have five problems - but they are five simpler and more specific problems.

Now we will need to break each of these five steps into smaller parts. Lets break (1.1) down into several sub-problems. In order to enter a theatre we have to walk up

to the theatre, then find the door, then wait in line to get in, then go through the door. We can write these a little better as:

refinement of enter theatre
1.1.1)walk up to theatre
1.1.2)find entrance doorway
1.1.3)wait in line to get to doorway
1.1.4)go through doorway

Now you may not agree with these latest steps. What if the box-office is outside the door? What if the people line up inside the theatre? The recipe has gotten more detailed and more specific. Try to refine the other four steps for yourself and see how they compare to mine. They probably don't match exactly and that's fine. Everyone breaks down a problem in a different way, and comes up with a different recipe to solve it. Programming is a truly creative process.

Here is my complete refinement for going to the movies:

1) See a movie

refinement of see a movie
1.1)enter theatre
1.2)buy ticket
1.3)enter viewing room
1.4)watch movie
1.5)leave theatre

refinement of enter theatre
1.1.1)walk up to theatre
1.1.2)find entrance doorway
1.1.3)wait in line to get to doorway
1.1.4)go through doorway

refinement of buy ticket
1.2.1)walk up to ticket counter

1.2.2)take out money ($3.50 tops)
1.2.3)state movie name (e.g. Hunt for Red October)
1.2.4)slide money through slot
1.2.5)receive ticket and change
1.2.6)get ticket ripped in half by the ticket-in-half-ripper person

refinement of enter viewing room
1.3.1)find correct viewing room within the cinerama quintaplex
1.3.2)find row where screen fills vision, no exit signs visible
1.3.3)move across row until within centre of viewing room
1.3.4)find closest empty chair
1.3.5)sit down

refinement of watch movie
1.4.1)talk with friends until projector turns on
1.4.2)watch previews
1.4.3)shut up
1.4.4)watch movie
1.4.5)wait for credits to finish and music to stop

refinement of leave theatre
1.5.1)stand up
1.5.2)unstick feet from floor
1.5.3)go outside through exit door

You could break each of these steps into several substeps. You could break step 1.1.4 "go through doorway" down into several motions. We will stop at three levels for this problem. We have broken down the problem to basic human conscious actions. The final number of refinements will depend on how complex the problem is. The more complex the problem is, the more steps it will take to break the problem into sufficiently simple steps that your audience can perform them.

You may have noticed at final registration for classes they now have a little chart that you follow along step by step as you go through the process. They have broken the big and general problem of "final registration" into a set of simpler steps for you to follow in order that will solve the problem of getting you registered for classes. Even the university administration can see the advantages inherent in this process.

Programming a computer to solve a problem involves a similar process, except

that the computer is the dumbest friend you have ever known. It will do exactly what you tell it, so you can not be ambiguous in your directions. You also must break down your problem to a level that the computer can handle on its own. This is when most people stop viewing computers as mysterious and unknowable, and start complaining about how 'stupid' they really are.

This ability to break down a problem statement down into steps is probably the most important thing to be learned in this class. Breaking down the problem forces you to understand the problem before you try to solve it. It is very tempting to just go right at the problem and start trying to fight the problem without thinking first, but it is always faster to slow down, sit back and think before you type.

We are going to start with a very simple problem expressed in English. We are going to break that problem down into a step by step English algorithm. We will then write these steps in the Pascal language so that the computer can understand them. When we have finished we will have a complete Pascal program.

Problem $\longrightarrow$ Algorithm $\longrightarrow$ Program

In everyday life we speak to each other in the language we are most familiar. For the sake of argument lets say it's English. When we speak English to one another we usually understand what each other is saying and if the sentences are ideas, orders or requests, we usually understand what it means and how to go about fulfilling the request. Unfortunately there are also instances when we do not communicate our orders effectively, perhaps because of a lack in the ability to speak the language or a lack in the specificity of the orders. For example in a recipe to bake a cake you may be asked to put in two eggs. Two people may interpret that in different ways. The obvious interpretation would be to crack the egg and empty its contents into the mixing bowl; an alternate interpretation is that we throw in the two eggs, shell and all. Speaking in a computer language so that the computer understands your intentions presents the same problems.

You have to treat the computer as a 3 year old child to get it to perform the tasks exactly as you intended. You must first be able to speak the computer language accurately because the computer is absolutely intolerant of incorrect grammar. And you must be able to break up the descriptions of a task you wish to perform into small enough parts so that the computer will understand without ambiguity. The rule to remember is: Computers never do what you **WANT** them to do; only what you **TELL** them to do. As Aldous Huxley said:

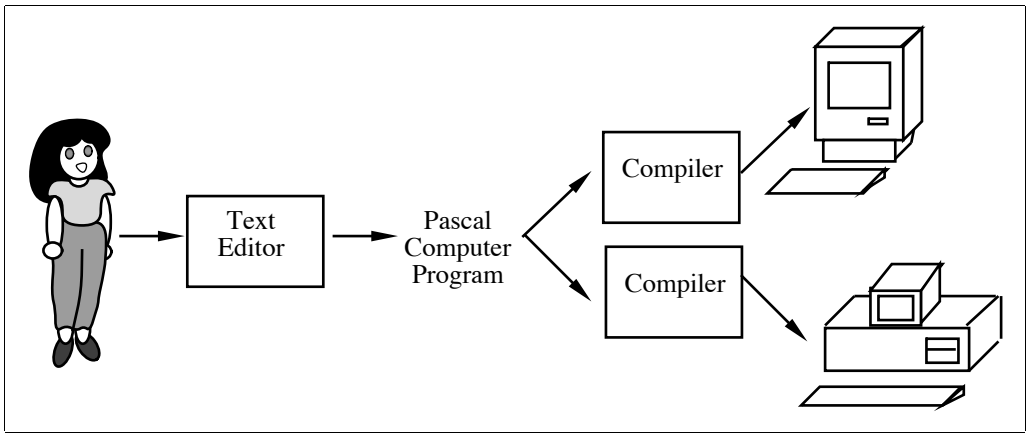


Figure 12.1: Portability of Pascal to different CPUs

> People always get what they ask for; the only trouble is that they never know, until they get it, what it actually is that they have asked for.

We have said that computers only know 0s and 1s. It would be rather tedious and very error-prone if we had to write programs with only 0s and 1s in this binary language. Fortunately each CPU on the market has its own **assembly language** which makes things a little better ...a little closer to English. **Pascal** and other **high level languages** such as **C**, **BASIC**, and **Lisp** make computer programs look even more like English. This makes the programs much easier to write, and much easier to understand. Even so, computer programming languages have a very precise grammar, and any deviation from it will cause the computer to reject what you write.

12.3 Portability

Programs written in high level languages such as Pascal are portable. I can write the same Pascal program on an IBM-PC or a Macintosh even though they have different CPUs. The code within the Pascal program does not depend on the brand of machine it is running on. Pascal programs must be translated into assembly language before they can be run. When this translation is performed they are translated into the specific assembly language of the CPU that will be used. This assembly language (**low level language**) version is no longer portable. It can only be used with that same type of CPU. Figure 12.1 illustrates.

12.4 Pascal

We are going to be using the computer language called Pascal in this class, which was named after the French mathematician Blaise Pascal. We are using it because it is a popular language at the moment in computer science curriculums, and it is very similar to most other computer languages you may encounter. It is also the language used in CSC 102, and this class is a prerequisite for that class.

We begin with a simple problem of teaching the computer how to add two numbers together and display the answer. This is a classic program as we all know that computers are good at arithmetic.

In plain English how is this problem expressed?

We may write:

I need to teach the computer how to add two numbers together and I would like to see the answer that the computer gets.

This is a clear and simple problem statement. It is important that you understand the problem before you attempt to solve it. Unless you fully understand the problem, you will never really know that what you are doing will solve it. This is a fact that applies to problem solving in general and not just computers. In Software Engineering, defining the problem occupies 30% of the effort of solving the problem.

The problem statement we have written above is fine for humans but far too imprecise for computers.Let us break the statement into smaller parts such as:

1. Give me two numbers.
2. Add them together.
3. Show the answer.

These smaller parts indicate a **stepwise refinement** of the problem statement. We should now attempt to exercise stepwise refinement to make the algorithm above more precise until finally we reach a stage where we can convert the idea directly into Pascal. A refinement to the above algorithm is:

1a. Give me a number.
1b. Give me another number.
2. Add them together.
3. Show the answer.

We break step 1 into two parts 1a, and 1b. You will notice that this looks a bit like an outline. We use letters and numbers to identify each step so we can see the refinement process. At this point, Step 2 is still somewhat imprecise. What do we mean by, Add THEM together? We therefore revise step 2 with a new refinement as follows:

1a. Give me a number.
1b. Give me another number.
2. Add the first number and second number together.
3. Show the answer.

Now, Step 3 seems a little imprecise. What do we mean by SHOW the answer? Let us rewrite the above as:

1a. Give me a number.
1b. Give me another number.
2. Add the first number and second number together.
3. Print the answer in step 2 onto the computer screen.

Since computers tend to be mathematical machines that often deal with symbols let us try to gradually move toward using symbols in our algorithm.

For example instead of saying: 1a. Give me a number. Let us say, Give me a number and call it NUMBERONE. Likewise we can call the second number NUMBERTWO. NUMBERONE and NUMBERTWO are good names for these symbols since we know that they are going to be numbers. Other good names could be FirstNumber and SecondNumber, or Number1 and Number2. So the algorithm now becomes:

1a. Give me a number and call it: NUMBERONE.
1b. Give me a number and call it: NUMBERTWO.
2. Add NUMBERONE and NUMBERTWO together.
3. Print the answer in step 2 onto the computer screen.

Let us also try to elaborate on step 2 and re-write it using another symbol THESUM. THESUM is a good name since we want to compute the sum of the two numbers. We could have called this NUMBERTHREE, or ANSWER but THESUM is much more specific. Let us use THESUM to contain the results of adding NUM-

BERONE and NUMBERTWO.

1a. Give me a number and call it: NUMBERONE.
1b. Give me a number and call it: NUMBERTWO.
2. Let THESUM = NUMBERONE + NUMBERTWO.
3. Print the answer in step 2 onto the computer screen.

Now we can rewrite step 3 so that our algorithm becomes:

1a. Give me a number and call it: NUMBERONE.
1b. Give me a number and call it: NUMBERTWO.
2. Let THESUM = NUMBERONE + NUMBERTWO.
3. Print the answer, THESUM, onto the computer screen.

At this point we can see that certain steps are becoming more precise while others are still somewhat vague. For example, step 1a and 1b says: Give ME a number and call it ...We don't want to use the word ME when we talk about computers so we use a term that sounds more like a computer term: READ. We want to read in values for these two numbers. But instead of just reading in the number, we should ask the user to give us a number. This is slightly more polite. So we can rewrite the steps as:

1ai. Ask user for NUMBERONE.
1aii. Read NUMBERONE.
1bi. Ask user for NUMBERTWO.
1bii Read NUMBERTWO.
2. THESUM = NUMBERONE + NUMBERTWO.
3. Print the answer, THESUM, onto the computer screen.

Similarly the third step: Print the answer ...can be more tersely expressed if we used WRITE. We want to write out the answer.We should also write out an appropriate message telling what this answer represents. The new refinement becomes:

1ai. Ask user for NUMBERONE.
1aii. Read NUMBERONE.
1bi. Ask user for NUMBERTWO.
1bii Read NUMBERTWO.
2. THESUM = NUMBERONE + NUMBERTWO.

3a. Write explanation.
3b. Write THESUM.

At this point our instructions are beginning to look more like a Pascal program. In fact in Pascal there are commands: READ and WRITE that does exactly what we want: to READ-in information (from the keyboard) and to WRITE-out information to the screen. With what we have developed so far, the grammar is still not quite correct Pascal so we need to re-write our steps using the Pascal grammar. Steps 1aii, 1bii and 3b may be re-written as follows:

1ai. Ask user for NUMBERONE.
1aii. Read (NUMBERONE);
1bi. Ask user for NUMBERTWO.
1bii Read (NUMBERTWO);
2. THESUM = NUMBERONE + NUMBERTWO.
3a. Write explanation.
3b. Write (THESUM);

Now we have to deal with writing out some explanatory text. Similar to the way we can Write (THESUM) we can just write out some plain text. Steps 1ai, 1bi, and 3a are re-written as follows:

1ai. Write ('Please enter the first number');
1aii. Read (NUMBERONE);
1bi. Write ('Please enter the second number');
1bii Read (NUMBERTWO);
2. THESUM = NUMBERONE + NUMBERTWO.
3a. Write ('The sum of the two numbers is');
3b. Write (THESUM);

Note the added parentheses and semi-colons after the instructions. At this point our problem is almost a Pascal program. It is "almost" a Pascal program because we need to make some changes to step 2. It is still not in the correct grammatical form (or **syntax**). We should rewrite step 2 as shown below, and at the same time we need to remove the step labels we have used so far because Pascal does not use them.

```
program AddEmUp;

var
   NumberOne, NumberTwo, TheSum: INTEGER;

begin
    WRITE('Please enter the first number');
    READ(NumberOne);
    WRITE('Please enter the second number');
    READ(NumberTwo);

    TheSum := NumberOne + NumberTwo;

    WRITELN('The sum of the two numbers is:', TheSum);
end.
```

Figure 12.2: Pascal Program

```
Write ('Please enter the first number');
Read (NUMBERONE);
Write ('Please enter the second number');
Read (NUMBERTWO);
THESUM = NUMBERONE + NUMBERTWO.
Write ('The sum of the two numbers is');
Write (THESUM);
```

Before this becomes a complete Pascal program we need to include some additional information in the program. These will be explained after we present the complete program in Figure 12.2.

12.5 Literate Programming

Stepwise Refinement is a technique often used by good Software Engineers. It has been greatly popularized by a technique called **Literate Programming**; invented by

professor Donald Knuth at Stanford University. It is an attractive means of programming because it forces the programmer to state the problem clearly and to concentrate on only one part of the problem at a time. Literate programming also forces programmers to thoroughly explain their program rather than just plunge in and write incomprehensible instructions. In fact a well written Literate Program should be as easy to read and follow as a good novel. In fact Knuth has published several books which are actually programs written in Literate Programming style, but because they have been written in this style, they have been sold as computer textbooks.

So in the spirit of Literate Programming we should explain (or comment or document) our program appropriately so that when we read it again at a later date we can still understand what it was supposed to do. We put a large comment at the top telling who wrote the program, when it was written, and what in general it was written to do. Important lines of code have comments written before them.

The new program with comments included is shown in Figure 12.3. Note: Text enclosed in {} pairs are recognized by Pascal as comments and not actual instructions. That is when Pascal works on the program it will ignore all the text that are enclosed in {}. This gives us a opportunity to insert "human" information there that makes our program more understandable without interfering with the normal interpretation of the program by the computer.

You may have noticed that we have indented the lines of the program. Along with comments and good variable names, indenting is used to make the program more readable to humans. When we write a paper in English we also indent our paragraphs to make them more readable.

12.6 Types of Errors

There are two different kinds of errors that are possible when writing programs. One is a **syntax error**, and the other is a **semantic error**. These programming errors are similar to their English equivalents. A syntax error is an error in grammar. The computer does not understand what you want it to do. It could be as simple as mistyping a letter, or misspelling a word. A semantic error is an error in meaning. You have given the computer instructions with proper grammar, but the machine does not do what you want it to do. You have told it to do the wrong thing and it is happily doing what you told it to do. Stepwise refinement can not eliminate syntax errors, but it can greatly reduce semantic errors, and the earlier you catch an error, the easier it is to fix it.

```
program AddEmUp;
{*********************************************}
{* Authors: Jason Leigh and Andy Johnson     *}
{* Date: 6/18/89 (last modification 4/29/90) *}
{*********************************************}
{* Description:                              *}
{* The computer will ask me for two numbers. *}
{* It will then add those                    *}
{* numbers together and show me the sum      *}
{*********************************************}

var
 NumberOne, NumberTwo, TheSum: INTEGER;

begin
 {print out welcoming message}
  WRITELN('Welcome to the number adding program.');

 {prompt the user for the numbers}
  WRITE('Please enter the first number');
  READ(NumberOne);
  WRITE('Please enter the second number');
  READ(NumberTwo);

 {calculate the sum of the two numbers}
  TheSum := NumberOne + NumberTwo;

 {print out the sum of the two numbers}
  WRITELN('The sum of the two numbers is:', TheSum);
end.
```

Figure 12.3: Complete Pascal Program

In English a sentence that is syntactically incorrect is "Jane the rifle Dick at shoots." There are several grammatical errors in that sentence. A sentence that is syntactically correct but semantically incorrect is "Colourless green ideas sleep furiously." Syntactically there is nothing wrong with this statement, but it doesnt make any sense: How can something be colourless and green? How can you sleep furiously? How can an idea be green, and how does it sleep?

12.7 Programming and You

The steps involved in converting our problem statement to a Pascal program is one of many possible solutions. Depending on the creativity of the programmer many different programs may appear, all solving the same problem in unique ways. There are no concrete steps that will always guide you from a problem statement to a final computer program. Successful translation requires a lot of practice and the more practice you get the better you will be able to perform this translation. It is like learning a foreign language. It requires a great deal of practice before you are able to speak the programming language fluently. But all of you have learned English, and it is far stranger than any computer language you will encounter.

12.8 Questions

1. Use stepwise refinement to refine the idea of going into a fast food restaurant and having lunch.
2. list 3 reasons why a person would want to write his own program.
3. Stepwise Refinement is used to
   (a) Break a problem down into smaller more manageable pieces
   (b) Convert a program from Pascal into Assembly Language
   (c) Compile a Pascal program
   (d) Create an executable version of a Pascal program
4. How does an algorithm relate to a program.?
5. What is the difference between a syntax error and a semantic error?

6. The language Pascal was named after

   (a) Bernie Pascal

   (b) Jean-Luc Pascal

   (c) Blaise Pascal

   (d) Luther Pascal

7. Develop an algorithm and a Pascal program for computing the value of Y in the following equation: $Y = MX + B$.

8. Develop an algorithm and a Pascal program for computing the average of your exam grades.

9. Develop an algorithm and a Pascal program to convert Fahrenheit to Celsius, or vice-versa.

Chapter 13

Some Pascal

We are now ready to actually talk about Pascal programming. You have used a word processor to type information into the computer. You have given specific commands to a database to perform specific functions. You have used formulas in a spreadsheet to do calculations. In the last chapter you learned how to break a problem down into smaller, simpler problems. Programming combines all these skills as you take a problem, break it down into specific instructions the computer understands, and then type it in to the computer using a text editor.

13.1 Line by Line

First we are going to show you the final version of the program we developed using stepwise refinement. We are now going to look at the individual lines of code and actually see what they do. The final version of the program is shown in figure 13.1.

Now my guess is that you understand what that program does, even if you do not know any Pascal. That is the kind of program that we want you to be able to write ...one that can be read and understood by people who do not know anything about programming.

Let's look at this program in a little more detail:

```
PROGRAM AddEmUp;
```

Every Pascal program must start with the word "program" and then the one-word program name and then a semi-colon. In this case the author of the program named

```
program AddEmUp;
{*********************************************}
{* Authors: Jason Leigh and Andy Johnson     *}
{* Date: 6/18/89 (last modification 4/29/90) *}
{*********************************************}
{* Description:                              *}
{* The computer will ask me for two numbers. *}
{* It will then add those                    *}
{* numbers together and show me the sum      *}
{*********************************************}

var
 NumberOne, NumberTwo, TheSum: INTEGER;

begin
 {print out welcoming message}
  WRITELN('Welcome to the number adding program.');

 {prompt the user for the numbers}
  WRITE('Please enter the first number');
  READ(NumberOne);
  WRITE('Please enter the second number');
  READ(NumberTwo);

 {calculate the sum of the two numbers}
  TheSum := NumberOne + NumberTwo;

 {print out the sum of the two numbers}
  WRITELN('The sum of the two numbers is:', TheSum);
end.
```

Figure 13.1: Complete Pascal Program

the program AddEmUp.

```
{*********************************************}
{* Authors: Jason Leigh and Andy Johnson     *}
{* Date: 6/18/89 (last modification 4/29/90) *}
{*********************************************}
{* Description:                              *}
{* The computer will ask me for two numbers. *}
{* It will then add those                    *}
{* numbers together and show me the sum      *}
{*********************************************}
```

The words on these lines are encased in the squiggly brackets '{' and '}'. Anything enclosed in squiggly brackets is called a comment. Comments are used to make the program more readable to a human who wants to know what your program does. The computer ignores all of your comments when it translates your program. The comment at the top of the program usually tells who wrote the program, when it was written, and what it does in general.

After the comments comes a blank line. The computer does not need it but it makes your code much easier to read if important parts of it are sectioned off using blank lines.

```
var
 NumberOne, NumberTwo, TheSum: INTEGER;
```

Here is where we declare what type each of our variables will be (var is short for variables.) A variable is a location in memory that is going to be used to store a value while the program is running. This place in memory is given a name (such as NumberOne.) We also give this location in memory a type (such as integer) which tells that location what kinds of things it can expect to be put there. In this program we set aside three memory locations and all of them will hold integers. We do not give any values to the variables at this point, we only declare their existence. It is important to give the variables appropriate names to make the program easier to read.

```
begin

end.
```

The 'main body' of the program, where the actual computing goes on, is contained between the words 'begin' and 'end.' It is important to note that after the end there is a period. Just as a Pascal program begins with the word 'program' it also must end with 'end.' You may notice that the text within the begin/end block is indented, as were the variables we declared. The computer also ignores all the indenting and all the white space you put into your program. When writing a paper we use paragraphs to break up our writing. We indent the first line of our paragraphs, and indent quotations even more. This indenting does not change the meaning of the words that we have written. It makes them easier to read. That is why we use indenting and white space in our Pascal programs. We want to make them easier to read.

```
{print out welcoming message}
 WRITELN('Welcome to the number adding program.');
```

The first line is contained in squiggly brackets so it is a comment. It tells us in English what is going to happen next. The second line uses the word 'writeln' and then there is a pair of parentheses, and finally a semi- colon. Writeln is short for "write line." Writeln is used to print text onto the monitor screen. What it prints is contained within the parenthesis. Writeln can be used to print out the values of variables, or just text. If text is to be written out it must be contained within single quotes (note not " [double quotes] or ' [apostrophe].) In this case the program will write out the phrase *Welcome to the number adding program.*

```
{prompt the user for the numbers}
```

Here we have another comment telling the reader what the program is going to do next.

```
WRITE('Please enter the first number');
READ(NumberOne);
```

Above we saw a writeln, and here we have a write. Both of them write out text onto the screen but the writeln moves down to the next line when it is finished ... like hitting the carriage return on a typewriter. The write writes out text but does not

hit the carriage return. This write statement is acting as a prompt. It is prompting the user of the program to type in a number. The next line uses the word read, a pair of parenthesis, and a semi-colon. Where 'write' is used to display information to the user, read is used to get information from the user. In this case we are going to read a value into the variable named NumberOne.

```
WRITE('Please enter the second number');
READ(NumberTwo);
```

These two lines behave in a similar way to the two lines above. The user is prompted to enter a second number, and the value that the user types in is stored in the variable NumberTwo.

```
{calculate the sum of the two numbers}
 TheSum := NumberOne + NumberTwo;
```

Now we are going to calculate the sum of the two numbers. The first line is a comment that tells us this. The second line is an assignment statement. ':=' is the assignment operator. What ':=' does is that it calculates the value on the right side and assigns that value to the variable on the left side. In this case the variable TheSum will be assigned to be the sum of NumberOne and NumberTwo. You can see here that the variable named TheSum is appropriately named since it is going to contain the sum. Now you can see why variables are called variables. TheSum, NumberOne, and NumberTwo do not have any values at this point. The values will be supplied by the user when the program is run. The contents of these variables will vary depending on what happens when the program is run.

```
{print out the sum of the two numbers}
 WRITELN('The sum of the two numbers is:', TheSum);
```

The two final lines within the main body are responsible for printing out the answer. The comment tells us that. Again we use a writeln to write out some information to the user. In this case we are writing out the text The sum of the two numbers is: and then we write out the value contained in the variable called TheSum.

13.2 Trace

Now you have seen what each line of the program does, so now lets look at what the program as a whole does. To do this we are going to use a technique called tracing. When we trace a program we are "running" it on a piece of paper in the same way that a computer would run it. In order to make what's going on a little clearer we are going to add some line numbers. These are shown in Figure 13.2

We start "running" the program at line 0. At this point we do not know what the values of the three variables are, so we mark them with a question mark. When we get to line 1 an introductory message is printed on the screen telling the user that the program is running. In line 2 Another message is printed prompting the user to enter a number. The values of all the variables are still unknown at this point. In line 3, the computer is waiting for the user to enter a number. Our imaginary user enters the number 4 in response to the computer's prompt. Now the value 4 is stored as the value of the variable named NumberOne. Line 4 prompts the user for the next number, and line 5 reads in the number that the user types. In this case our imaginary user types in the number 11. In line 6 the computer calculates the sum of the two numbers and stores the value 15 in the variable TheSum. Line 7 prints out the value in the variable TheSum along with an appropriate message. Then the program is done. See Figure 13.3.

You have now seen two ways to put a value into a variable. You can either read in a value using a READ statement, or use the assignment statement :=. The WRITELN statements do not change the value within the variables.

13.3 Semi-Colons

You are probably wondering why there are so many semi-colons in Pascal when you hardly ever see them in English. Pascal does not look for carriage returns or white space to end a statement. The semicolon is used to separate statements. In English we use periods to separate our sentences. We can write sentences that are longer than one line. We can write short sentences. The end of a line does not signal the end of an English sentence, a period does. The same is true for Pascal. The end of a line does not signal the end of a Pascal Statement, a semi-colon does.

```
program AddEmUp;
{*********************************************}
{* Authors: Jason Leigh and Andy Johnson     *}
{* Date: 6/18/89 (last modification 4/29/90) *}
{*********************************************}
{* Description:                              *}
{* The computer will ask me for two numbers. *}
{* It will then add those                    *}
{* numbers together and show me the sum      *}
{*********************************************}

var
 NumberOne, NumberTwo, TheSum: INTEGER;

{0} begin
    {print out welcoming message}
{1}  WRITELN('Welcome to the number adding program.');

    {prompt the user for the numbers}
{2}  WRITE('Please enter the first number');
{3}  READ(NumberOne);
{4}  WRITE('Please enter the second number');
{5}  READ(NumberTwo);

    {calculate the sum of the two numbers}
{6}  TheSum := NumberOne + NumberTwo;

    {print out the sum of the two numbers}
{7}  WRITELN('The sum of the two numbers is:', TheSum);
end.
```

Figure 13.2: Complete Pascal Program with Line Numbers

Line #	Number One	Number Two	The Sum	What is Printed On The Screen	User Types
0	?	?	?		
1	?	?	?	Welcome to the ... program.	
2	?	?	?	Please enter the first number	4
3	4	?	?		
4	4	?	?	Please enter the second number	11
5	4	11	?		
6	4	11	15		
7	4	11	15	The sum of the two numbers is: 15	
done					

Figure 13.3: Tracing the Program

13.4 Style

It is possible to write a very different looking Pascal program that will do exactly the same job as the program shown in Figure 13.1. This version is shown in Figure 13.4. As far as the user is concerned the programs are exactly the same but this program is very unreadable and this style should not be emulated.

What is so bad about it?

1. meaningless program names
2. meaningless variable names
3. no indenting
4. no white space
5. no comments
6. more than one statement per line

It is very important for you to write readable Pascal programs, if only for the reason that it makes your teacher happy, and more likely to give you a good grade.

```
PROGRAM test; VAR X, Y, Z: INTEGER; BEGIN
WRITELN('Welcome to the number adding program.');
WRITE('Please enter the first number');
READ(X); WRITE('Please enter the second number');
READ(Y); Z := X + Y;
WRITELN('The sum of the two numbers is:', Z); END.
```

Figure 13.4: Bad Style

Mostly it is a question of style. We want you have a good programming style. Programs should be readable, and easy to follow. Just getting the program to work is not enough. In fact its only about half the challenge.

13.5 Variables

In mathematics a variable is something like "X" or "Y" and they are used in equations such as Y = 5X + 2. "X" and "Y" are variables because they do not have a set value - they are variable. In computer languages a variable is the name of a certain location in the RAM that is used to store a value. The value stored at this location could be 12 or it could be the letter 'C' so the contents can vary. Computer memory can be compared to a wall of mail-boxes. Each one has a unique name, and their purpose is to store something.

Up in the VAR section is where we **declare** our variables. We tell the computer that we need some space reserved in memory, and we are going to refer to that space by a certain name. We also tell the computer what kinds of things we are going to put in that space. Declaring a variable is like writing your name on your mailbox. You have claimed that space and given it a name.

Within the body of the program is where we use the variable. Declaring a variable does not give it any specific value. This is why it is a good idea to **initialize** your variables to give them a specific starting value. When you write your name on the mailbox, you will very likely want to open it up and take out any old mail that has been left over from the previous tenant. Similarly in a computer program you do not know what's been left in the memory that you have just claimed for your variable. Its best to clean it out and set it to a specific value to avoid problems. The first

Space Shuttle flight was delayed several weeks because someone did not initialize their variables.

Like your mailbox there are only a few things you can do with a variable. You can look and see whats inside, or you can put something new inside. A variable can only hold one thing so its like a very small mailbox. If you try to put another value into a variable it replaces the old one. So if the variable "Temperature" already contains the value 60 and you issue the command "Temperature := 75;" the variable "Temperature" now contains the value 75. The 60 is gone, replaced, no more. If the letter carrier comes to put a letter in your box and he finds you already have a letter there he kindly rips up the old letter and substitutes the new one in its place.

In the first sample program we made all three variables of the type 'integer.' There are three other major types of variables you need to be aware of: real, boolean, char.

13.5.1 Integer

Integers are the whole numbers between 32768 and -32768. Now this might be a little limiting. Integers do not include fractions, and they do not include large numbers. integers are good for everyday type calculations where the numbers do not get very large.

13.5.2 Real

Reals are rational numbers. They have a whole part and a decimal part. Real numbers have a much wider range than the integers, but real numbers are not exact. We said before that computers store everything as a sequence of 0s and 1s. A number such as π or 1/3 can not be represented with a finite number of 0s and 1s, so the number stored in a real variable is not exact.

13.5.3 Char

Integers and Reals pretty much cover the numbers, but computer programs can deal with more than numbers. Computers can also deal with letters. A variable that holds a single letter is of type char (short for character.) A variable that holds a sequence of letters is a string. Characters include more than letters, they also include the single digits, and the punctuation marks. Any single character that you can type with a typewriter is a character in Pascal.

13.5.4 Boolean

Since computers deal with 1s and 0s,on and off, yes and no, there is a special variable type to handle these. It is called Boolean, named after logician George Boole. Boolean variables are either true or false ... that's it.

13.6 Reserved Words

When we declare a variable we give it a name and a type. Now we have said that we want to give variables good long names, but there are some names we can not use for variables. Like many programming languages, Pascal has a set of reserved words. These reserved words have special meanings in the language so they are not available to the programmer. A listing of Pascal's reserved words is given below:

and	array	begin	case	const
div	do	downto	else	end
file	for	function	goto	if
in	label	mod	nil	not
of	or	packed	procedure	program
record	repeat	set	then	to
type	until	var	while	with

13.7 Arithmetic

In the first sample program there was a line that read:

```
{calculate the sum of the two numbers}
 TheSum := NumberOne + NumberTwo;
```

We used the "+" symbol to stand for addition, just like in arithmetic. We use the "-" (dash) for subtraction. Now computers do not use $\times$ and $\div$ for multiplication and division. Instead "*" (asterisk) means multiplication and "/" (slash) means division. All of the standard rules of algebra apply in Pascal, including the precedence rules, but remember that := (colon equals) is not an equal sign, but an assignment statement.

For example:

```
ForceObject := MassOfObject * AccelerationOfObject;
```

is a Pascal statement that takes the value of the variable MassOfObject and multiplies it by the value of the variable AccelerationOfObject and then stores the result into the variable ForceObject. Whatever value that was in ForceObject before is now gone. It has been replaced by the new value.

If we had a variable which stored a length in inches and we wanted to convert that into the number of centimeters we could do it as follows:

```
NumberCentimeters := NumberInches * 2.54;
```

If we wanted to do the reverse calculation and convert the number of centimeters into the number of inches we could not use this same line of code. We would need to use the following line of code:

```
NumberInches := NumberCentimeters / 2.54;
```

Now here we do have a bit of a problem. You see adding, subtracting, or multiplying integers will always give you another integer. This is not true of division. Pascal assumes that the "/" operator will produce a real number as a result. Now what happens if we try to put a real value such as 12.5 into an integer variable. It is like trying to put a round peg into a square hole. It won't fit, and Pascal will tell you. It will give you an error message. You can however assign an integer value such as -34 to real variable as it will simply add on a ".0" to the end of it.

Now how about this one:

```
Counter := Counter + 1;
```

This statement is meaningless in arithmetic since $X \neq X + 1$. But as we said before := is an assignment statement. What is on the right side? "Counter + 1;" OK, we can calculate that value. Then we assign that value to the variable on the left hand side. In this case that variable also happens to appear on the right hand side. No problem. This statement increments the value of Counter by 1.

13.8 Write/Writeln

Above we had a brief discussion on the difference between WRITE and WRITELN. Since this often causes confusion we will give a few examples. Remember that WRITE prints something onto the screen and then waits there. Writeln does a write and then appends a carriage return to the end.

```
WRITELN('A', 'B', 'C');
WRITE('d', 'e', 'f');
WRITE('G', 'H', 'I');
WRITELN('j', 'k', 'l');
WRITELN('M', 'N', 'O');
```

creates the following output:

```
ABC
defGHIjkl
MNO
```

As there is a WRITE and WRITELN for outputting information to the user; there is a READ and READLN for inputting information from the user. Just as WRITE will print out information and wait on the same line, READ will read in information and wait on the same line. Just as WRITELN goes to the beginning of the next line after writing out its information, READLN will read in information and then go to the beginning of the next line.

13.9 Some Pascal and You

And that is some Pascal. What you have learned so far will allow you to write some fairly simple, straightforward programs (and those are the best kind.) As with any language, you do not become fluent unless you practice it.

13.10 Questions

1. What is the difference between declaring a variable and initializing it? What will happen if we don't declare a variable? What will happen if we don't initialize it?

2. Why is it important for you to put comments into your code?
3. If you try to assign a real value to an integer variable
   (a) the fractional part is ignored
   (b) the fractional part is rounded to the nearest integer
   (c) the integer variable will be converted to a real variable
   (d) you will get an error message
4. Writeln is different from Write in that Writeln
   (a) goes to the next line then does a write
   (b) does a write then goes to the next line
   (c) goes to the next line, does a write, then goes to the next line
   (d) only prints out integers while write prints out real numbers
5. Why is it important to put comments in your program?
6. An integer is a
   (a) any number
   (b) positive whole number
   (c) positive whole number or zero
   (d) positive or negative whole number or zero
7. Indenting is used in a program so that
   (a) the computer knows the order to execute the statements
   (b) the computer can create an application
   (c) the computer can read the program easier
   (d) the program is made more readable to a human
8. All Pascal programs end with a
   (a) :
   (b) ;
   (c) .
   (d) ..

Chapter 14

More Pascal

There are three different orders in which statements will be executed: Sequentially, Conditionally, and Repititionally (yes, I know there is no such word as repititionally, but I like it.) You have already seen sequential programs. In them each line is processed in turn, and then the next line is processed and so on. When statements are executed Conditionally they may or may not be executed depending on whether the condition is met. When statements are executed repititionally they may be executed more than one time. All three different types of control are necessary to write good programs. Figure 14.1illustrates.

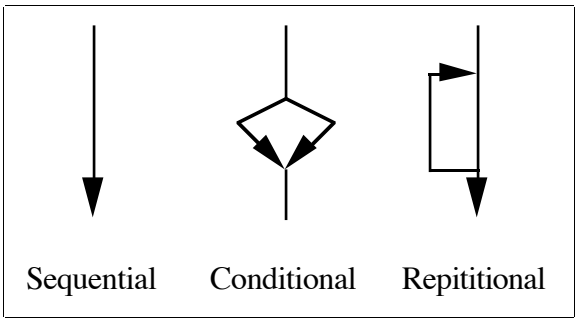


Figure 14.1: Different Statement Orders

14.1 If

The most common conditional statement in Pascal is the "IF" statement. It looks like this:

```
IF condition THEN
  BEGIN
    {do if condition is true}
  END;
```

or it can look like this

```
IF condition THEN
  BEGIN
    {do if condition is true}
  END
ELSE
  BEGIN
    {do if condition is false}
  END;
```

Within the BEGIN END block of the IF statement we can have any Pascal statements that we wish. We can do arithmetic, read in values, write out values, or even have other IF statements.

Let's say we want to write a Pascal program to help us plan our daily exercise. We want to give the program the current outside temperature, and the computer will tell us what we should do. We can set up the following table:

$80 < \text{Temp}$	swimming
$60 < \text{Temp} \leq 80$	tennis
$40 < \text{Temp} \leq 60$	baseball
otherwise	stay inside

Now assuming we have the outside temperature stored in the variable Temp we can handle the first case by writing:

```
IF Temp > 80 THEN
  BEGIN
    WRITELN('Go Swimming');
  END;
```

If Temp is larger than 80 we will get the message *Go Swimming*. "Temp > 80" is the condition that is being evaluated. If this is true then the message *Go Swimming* is printed. If the condition is not true then the message will not be printed. Now we need to add in the condition for playing tennis. We could write the following.

```
IF Temp > 80 THEN
  BEGIN
    WRITELN('Go Swimming');
  END;
IF Temp > 60 THEN
  BEGIN
    WRITELN('Play Tennis');
  END;
```

Unfortunately, this won't work. If Temp is 75 we get the correct message, but what happens if the temperature is 90. Then we get BOTH messages printed. That's not what we wanted. Here is one way to avoid this problem.

```
IF Temp > 80 THEN
  BEGIN
    WRITELN('Go Swimming');
  END;
IF (Temp > 60) AND (Temp <= 80) THEN
  BEGIN
    WRITELN('Play Tennis');
  END;
```

We have made the second condition more explicit, so the *Play Tennis* message will only be printed if Temp is greater than 70 AND less than or equal to 80. The parentheses are needed now that we have multiple conditions. The rest of the code looks basically the same:

```
IF Temp > 80 THEN
  BEGIN
    WRITELN('Go Swimming');
  END;
IF (Temp > 60) AND (Temp <= 80) THEN
  BEGIN
```

```
    WRITELN('Play Tennis');
  END;
IF (Temp > 40) AND (Temp <= 60) THEN
  BEGIN
    WRITELN('Play Baseball');
  END;
IF Temp < 40 THEN
  BEGIN
    WRITELN('Stay home and watch videos');
  END;
```

The entire program would look something like Figure 14.2.

14.2 If Then Else

Figure 14.3 shows another way to avoid the problem using the IF-THEN-ELSE version of the IF statement.

Now that probably looks a bit messier to you, but it will do the same thing. Note that there is never a ";" before an ELSE. Semi-colons are used to separate one statement from another. The ELSE is still part of the IF-THEN-ELSE statement. You would not put a semi-colon in the middle of the word BEGIN(BEG;IN), so you do not put one before an ELSE.

14.3 Loops

There are three different types of repititional statements in Pascal. These are generally called "loops" since they can loop back and perform the same job over and over. We will deal with two of the three: the FOR loop and the WHILE loop.

14.3.1 For loop

The FOR loop looks like this:

```
FOR Counter-variable := start to finish Do
  BEGIN
    {do this finish - start + 1 times}
  END;
```

```
PROGRAM Exercise;
{ Author: Andy Johnson     Date: 4/29/90
  The computer will tell me what exercise to perform
      depending on the outside temperature }

VAR
    Temp : INTEGER; {the outside temperature}

BEGIN
{print out welcoming message}
    WRITELN('Welcome to the exercise decision program.');

{get the outside temperature}
    WRITELN('What is the temperature in Fahrenheit?:');
    READ(Temp);

{decide on what activity to perform}
    IF Temp > 80 THEN
     BEGIN
        WRITELN('Go Swimming');
     END;
    IF (Temp > 60) AND (Temp <= 80) THEN
     BEGIN
        WRITELN('Play Tennis');
     END;
    IF (Temp > 40) AND (Temp <= 60) THEN
     BEGIN
        WRITELN('Play Baseball');
     END;
    IF Temp < 40 THEN
     BEGIN
        WRITELN('Stay home and watch videos');
     END;
END.
```

Figure 14.2: Complete Pascal Exercise Program

```
PROGRAM Exercise2;
{ Author: Andy Johnson     Date: 4/29/90
  The computer will tell me what exercise to perform
      depending on the outside temperature}
VAR
    Temp: INTEGER; {the outside temperature}
BEGIN
{print out welcoming message}
    WRITELN('Welcome to the exercise decision program.');

{get the outside temperature}
    WRITELN('What is the temperature in Fahrenheit?:');
    READ(Temp);

{decide on what activity to perform}
IF Temp > 80 THEN
   BEGIN
      WRITELN('Go Swimming');
   END
ELSE
   BEGIN
      IF Temp > 60 THEN
         BEGIN
            WRITELN('Play Tennis');
         END
      ELSE
         BEGIN
            IF Temp > 40 THEN
               BEGIN
                  WRITELN('Play Baseball');
               END
            ELSE
               BEGIN
                  WRITELN('Stay home and watch videos');
               END;
         END;
   END;
END.
```

Figure 14.3: Complete Pascal Exercise Program w/ IF-THEN-ELSE

The WHILE loop looks like this:

```
WHILE condition Do
  BEGIN
    {do while condition is true}
  END;
```

Let's say we would like to write the word "hello" on the screen 5 times like this:

```
hello
hello
hello
hello
hello
```

Now we could just write a simple program to do the job as shown in Figure 14.4. But there is a better way, using a FOR loop as shown in Figure 14.5. Why is this better? Well, what happens if you need a program to print hello 50 times, or 500 times. It is much easier to change the 5 to a 500 in program HelloFor, than to type in 495 more writelns into program Hello. The program is much easier to modify. In this program the variable Counter is used as the index for the FOR loop. Counter is set to 1 the first time through the loop and then after executing the code within the loop, counter is automatically incremented by one. This process continues until Counter exceeds the upper limit of the FOR loop. In this case, Counter takes on values 1, 2, 3, 4, and 5.

There are some restrictions on the FOR loop. The counter-variable must be an integer as well as the starting value and the ending value. In program HelloFor we declared Counter to be an integer.

What if we wanted a program to print out the numbers 5 to 10 in a column like this:

```
5
6
7
8
9
10
```

We could certainly write a program with 6 writeln statements to do the job, but

```
PROGRAM Hello;
{This program will write out hello 5 times in a column}

BEGIN
  WRITELN('hello');
  WRITELN('hello');
  WRITELN('hello');
  WRITELN('hello');
  WRITELN('hello');
END.
```

Figure 14.4: Simple Hello Program

```
PROGRAM HelloFor;
{This program will write out hello 5 times in a column}

VAR
  Counter: INTEGER; {counter for the FOR loop}

BEGIN
  FOR Counter := 1 TO 5 DO
    BEGIN
      WRITELN('hello');
    END;
END.
```

Figure 14.5: Better Hello Program

```
PROGRAM FiveTen;
{This program will write out the numbers between 5 and 10 inclusive}

VAR
  Index : INTEGER;  {counter for the FOR loop}
  Start, Finish : INTEGER {start and finish for the FOR loop}

BEGIN
  Start  := 5;  {initialize start}
  Finish := 10; {initialize finish}

  FOR Index := Start TO Finish DO
    BEGIN
       WRITELN(Index);
    END;
END.
```

Figure 14.6: Counting from 5 to 10 Program

again, this would be hard to modify. We can use a FOR loop to this as shown in Figure 14.6.

What if we wanted to print the numbers in the opposite order, that is start with 10 then 9 then 8 down to 5. Program TenFive in Figure 14.7 shows you how to do this.

14.3.2 While loop

The WHILE loop is a more general loop that the FOR loop, so any FOR loop can be expressed as a WHILE loop. Figure 14.8 shows the program from Figure 14.5 using a WHILE loop. Figure 14.9 shows the program from Figure 14.6 using a WHILE loop.

A WHILE loop will keep looping as long as the condition remains true. In program HelloFor the condition is "Counter <= 5" so as long as Counter is less than or equal to five, the code within the loop will be executed. Before we get to the loop we set Counter equal to 1. Since 1 is less than 5 we enter the loop. Within the loop "hello"

```
PROGRAM TenFive;
{This program will write out the numbers between 10 and 5 inclusive}

VAR
  Index : INTEGER;  {counter for the FOR loop}
  Start, Finish : INTEGER {start and finish for the FOR loop}

BEGIN
  Start := 10;   {initialize start}
  Finish := 5; {initialize finish}

  FOR Index := Start DOWNTO Finish DO
    BEGIN
       WRITELN(Index);
    END;
END.
```

Figure 14.7: Counting from 10 to 5 Program

```
PROGRAM HelloWhile;
{This program will write out hello 5 times in a column}

VAR
  Counter : INTEGER;  {counter for the WHILE loop}

BEGIN
  Counter := 1;      {initialize counter}
  WHILE Counter <= 5 DO
    BEGIN
       WRITELN('hello');
       Counter := Counter + 1; {increment counter}
     END;
END.
```

Figure 14.8: Hello Program Using WHILE Loop

is printed and Counter is incremented to 2. 2 is less than 5 so we go through the loop again. "hello" is printed and Counter is incremented to 3. $3 \leq 5$ so we go through the loop again. "hello" is printed and Counter is incremented to 4. $4 \leq 5$ so we go through the loop again. "hello" is printed and Counter is incremented to 5. $5 \leq 5$ so we go through the loop again. "hello" is printed and Counter is incremented to 6. 6 is greater than 5 so we do not repeat the loop again.

The FOR loop automatically incremented Counter and Index. The WHILE loop does not. We must increment any counters that we choose to use. One of the dangers of a WHILE loop is to forget to increment the counter as shown in the following program. This program will keep printing "hello" on the screen until the computer is turned off. This is called an infinite loop. Program Infinite in Figure 14.10 illustrates.

Why does it go into an infinite loop? Because Index is set to one, and then it is never modified. Since $1 \leq 10$ the program will continue to loop. The looping condition will never become false.

The condition of the while loop does not have to use integers. It can involve reals, or characters or booleans, or any combination of them. Any expression that evaluates to either true or false can be used as the condition of a while loop.

```
PROGRAM FiveTen2;
{This program will write out the numbers between 5 and 10 inclusive}

VAR
  Index : INTEGER;  {counter for the WHILE loop}
  Start, Finish : INTEGER {start and finish for the WHILE loop}

BEGIN
  Start  := 5;  {initialize start}
  Finish := 10; {initialize finish}

  Index := Start;          {initialize counter}
  WHILE Index <= Finish DO
    BEGIN
       WRITELN(Index);
       Index := Index + 1;  {increment counter}
    END;
END.
```

Figure 14.9: Counting from 5 to 10 using a WHILE Loop

```
PROGRAM Infinite;
{This program will go into an infinite loop. This is not
a good thing for a program to do}

VAR
  Index : INTEGER;  {counter for the WHILE loop}

BEGIN
  Index := 1;          {initialize counter}
  WHILE Index <= 10 DO
    BEGIN
       WRITELN('hello');
       {forget to increment counter}
    END;
END.
```

Figure 14.10: Infinite Loop Program

```
PROGRAM Response;
{This program will ask the user if it should keep going}

VAR
Response: CHAR;  {sentinel for the WHILE loop}

BEGIN
  Response := 'Y';   {initialize Response}
  WHILE (Response = 'Y') OR (Response = 'y') DO
    BEGIN
      WRITELN('hello');
      WRITE('Do it again?(Y/N)');
      READLN(Response);
    END;
  WRITELN('I am done');
END.
```

Figure 14.11: WHILE Loop Under User Control

Program Response in Figure 14.11 will print "hello" and then ask the user whether it should do it again. If the user types in a capital Y or a lower case y the program will do it again. The program will keep going through this sequence until the user types something other than Y or y.

Note that the condition has = signs not :=. This is because := assigns a value, where = checks for equality.

In the above program, we have a compound condition for the while statement using the connective OR. The while condition will be true if either the first condition or the second condition is true or both are true. As well as OR, you can also use AND and NOT.

14.3.3 Logic

You might remember logical rules in Figure 14.12 from High-school. They come in rather handy in Pascal. You might have noticed that the preceding programs used

True AND True → True	True OR True → True
True AND False → False	True OR False → True
False AND True → False	False OR True → True
False AND False → False	False OR False → False
NOT True → False	
NOT False → True	

Figure 14.12: Rules of Logic

=	=
≠	<>
≤	<=
≥	>=
<	<
>	>

Figure 14.13: Mathematics VS Pascal

'<=' instead of '≤.' Since certain mathematical symbols are not available on the keyboard so Pascal has some substitutes as shown in Figure 14.13. You could have a complicated while condition like the following:

```
Counter := 1;
Response := 'Y';
WHILE (counter <= 5) AND NOT ((Response = 'N') OR (Response = 'n')) Do
  BEGIN
    WRITELN('hello');
    WRITE('Do it again?(Y/N)');
    READLN(Response);
    Counter := Counter + 1;
  END;
```

In the above program "hello" will be printed at most 5 times, but if the user does types a "N" or "n" to stop it, it will stop sooner. "hello" will be printed as long as:

(Counter $\leq$ 5) AND NOT ((Response = 'N') OR (Response = 'n'))

This can be seen as a collection of true and false clauses.

True/False AND NOT ((True/False) OR (True/False))

Depending on the values of the individual clauses the overall expression will be either TRUE or FALSE.

For example, if Counter = 4 and Response = "Y" then we have:

True AND NOT ((False) OR (False))

In order to simplify this we do parenthesis first, then NOT, then AND, and finally OR. These are the same simple logic rules that you learned about in high-school. In this case we have parenthesis so we start there:

False OR False $\longrightarrow$ False so we simplify to:

True AND NOT (False)

NOT False $\longrightarrow$ True so we simplify to:

True AND True

True AND True $\longrightarrow$ True so we simply end up with:

True

Since the while condition is true, the while loop would be executed for that set of values. The while condition can always be simplified down to either TRUE (the loop will execute again) or FALSE (the loop will not execute again.)

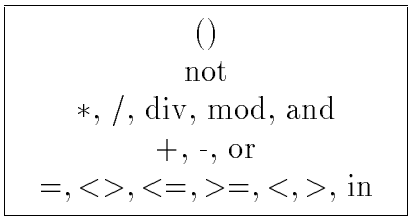


Figure 14.14: Order of Precedence

14.3.4 Precedence

At this point we have introduced several operators, so we should show their precedence order. You may remember that $5 \times 4 + 3 = 23$ not 35 because $\times$ has a higher precedence than +. Figure 14.14 shows the complete list in decreasing order of precedence.

14.4 GOTO

Some computer languages (including Pascal) have another way of ordering statements. These languages have what is called a GOTO statement. This statement allows the program to GOTO some other part of the program. The use of GOTOs is considered very bad programming practice. Programs with GOTOs are very hard to figure out, because the program is jumping from place to place. We have said that the goal of literate programming is to make programs readable to those who know nothing about computers or programming. If a program with GOTOs is confusing to someone who KNOWS about computers and programming, then it will be incomprehensible to people who know even less. When a programmer needs to use a GOTO, it shows that there was not enough work done in the stepwise refinement phase.

14.5 Procedures

Sometimes our programs get very long and it is bothersome to have to read page after page of code trying to find a certain section. Procedures solve this problem. Procedures break up the code into segments. Procedures embody the idea of stepwise refinement as each procedure contains a refinement of the overall program, and the

lines of code within the procedure refine the idea of the procedure. Figure 14.15 contains the program that we generated from stepwise refinement.

Here is the initial refinement:

1. Give me two numbers.
2. Add them together.
3. Show the answer.

We will use this initial refinement to break the program into procedures as shown in Figure 14.16.

Now we have broken the program into three procedures. The procedures are **declared** after the variables and look like little programs. The procedures are **called** from within the main body by giving the procedure's name.

```
PROCEDURE GiveMeTwoNumbers;
{prompt the user for the numbers}
  begin
    WRITE('Please enter the first number');
    READ(NumberOne);
    WRITE('Please enter the second number');
    READ(NumberTwo);
  end;
```

Here is the declaration of the GiveMeTwoNumbers procedure. It has an appropriate name, and a comment to tell us what this procedure does. You can see the similarities between a program and a procedure. The only difference is that the word PROGRAM is replaced by the word PROCEDURE and the "end." is replaced by an "end;" since there can only be one "end." in the program. Within the procedure there is a BEGIN-END block just like in a program. Anything that can be written in a program can be written in a procedure.

```
GiveMeTwoNumbers;
```

Procedure GiveMeTwoNumbers is activated (or called) by giving its name followed by a semi-colon. When a procedure is called, its statements are executed. After the procedure is finished, control is passed to the next line in the main program.

Procedures help make the code more readable. When you want to find the main body of the program all you have to do is go to the very end of the code and there

```
program AddEmUp;
{*********************************************}
{* Authors: Jason Leigh and Andy Johnson     *}
{* Date: 6/18/89 (last modification 4/29/90) *}
{*********************************************}
{* Description:                              *}
{* The computer will ask me for two numbers. *}
{* It will then add those                    *}
{* numbers together and show me the sum      *}
{*********************************************}

var
 NumberOne, NumberTwo, TheSum: INTEGER;

begin
 {print out welcoming message}
  WRITELN('Welcome to the number adding program.');

 {prompt the user for the numbers}
  WRITE('Please enter the first number');
  READ(NumberOne);
  WRITE('Please enter the second number');
  READ(NumberTwo);

 {calculate the sum of the two numbers}
  TheSum := NumberOne + NumberTwo;

 {print out the sum of the two numbers}
  WRITELN('The sum of the two numbers is:', TheSum);
end.
```

Figure 14.15: ADDEMUP Pascal Program

```
program AddEmUp;
{*********************************************}
{* Authors: Jason Leigh and Andy Johnson     *}
{* Date: 6/18/89 (last modification 4/29/90  *}
{* This version has procedures               *}
{*********************************************}
var
  NumberOne, NumberTwo, TheSum: INTEGER;

PROCEDURE GiveMeTwoNumbers;
{prompt the user for the numbers}
  begin
    WRITE('Please enter the first number');
    READ(NumberOne);
    WRITE('Please enter the second number');
    READ(NumberTwo);
  end;

PROCEDURE AddThemTogether;
{calculate the sum of the two numbers}
  begin
    TheSum := NumberOne + NumberTwo;
  end;

PROCEDURE ShowTheAnswer;
{print out the sum of the two numbers}
  begin
    WRITELN('The sum of the two numbers is:', TheSum);
  end;

begin
{print out welcoming message}
  WRITELN('Welcome to the number adding program.');

  GiveMeTwoNumbers;
  AddThemTogether;
  ShowTheAnswer;
end.
```

Figure 14.16: ADDEMUP Pascal Program with Procedures

it is. You can then look at the procedure names and go right to the procedure that you are interested in.

14.6 Numbers

When Pascal prints out a number it will try to print that number out using the most compact representation possible. This will usually mean scientific notation where 1200 would be written 1.2E+3. There are times when we would like to specify how the number will be printed out, especially if we are trying to set up columns of information. Pascal allows us to do this, though the format is slightly different for integers and reals.

When we are writing out an integer variable we can write it like this:

```
Writeln( NumberOne:5);
```

That is: VARIABLE:NUMBER. The number tells how many places on the screen to reserve for the value of the variable. The value of the variable will be right justified within this number of places. If NumberOne contains the value 13, the following would be printed:

$\underbrace{\_\_\_13}_{5}$

When we are writing out a real variable we can write it like this:

```
Writeln( Fraction:8:2);
```

That is: VARIABLE:TOTAL:FRACTION. The total tells how many places on the screen to reserve for the value of the variable, and the fraction tells how many places are after the decimal point. The value of the variable will be right justified within this number of places. If Fraction contains the value 124.6, the following would be printed:

$\underbrace{\_\_124.\overbrace{60}^{2}}_{8}$

14.7 Even More Pascal

There is much, much more to Pascal than what has been mentioned here, but a decent discussion of Pascal requires a book in itself. The concepts given here are valid for almost all higher level languages, and once you understand one high level language it is very easy to work with any of the others.

14.8 Questions

1. What are the advantages of using procedures in a program?
2. Why is a 'GOTO' considered bad programming practice?
3. What is an 'infinite loop'?
4. The code within a WHILE loop will be executed
   (a) as long as the while condition is true
   (b) as long as the while condition is false
   (c) once
   (d) never
5. The counter in a FOR loop
   (a) must be a character
   (b) must be a real number
   (c) is automatically incremented by one each time through the loop
   (d) must be incremented by the user

Chapter 15

Programming Languages

In this class you will only be working with the Pascal language, but there are many other languages out there that you will encounter if you wish to keep writing your own programs. Many different "higher-level" (or more English-like) computer languages have been written since the mid-50's. Some of the earliest languages are still widely in use today, as are some of the newer languages. Here we are going to write the same program in several different computer languages so you can see the similarities and the differences between them.

Each of these programs was written to calculate N! ("N factorial.")
For some value of N, N! $= N \times (N-1) \times (N-2) \cdots 3 \times 2 \times 1$.
For example, if N $= 5$ then N! $= 5 \times 4 \times 3 \times 2 \times 1 = 120$.

15.1 Pascal Version

Pascal was developed in the late 60's by Niklaus Wirth. In the 80's it has been the major language used to teach programming skills in universities. As you have seen in class, many personal computers have a version of Pascal available for them.

```
PROGRAM factorial;
{This Pascal program will prompt the user for a positive integer}
{and it will then calculate and return the factorial of that number}
{i.e. given N, the program calculates N!}

VAR
  number: integer;  {user input number}
  counter: integer;
  theFactorial: real;  {factorial of user number}

BEGIN
{get number from user}
  writeln('Welcome to the factorial calculator');
  write('Please enter a positive integer number:');
  readln(number);

{calculate the factorial}
  theFactorial := 1;

  FOR counter := 1 TO number DO
    BEGIN
      theFactorial := theFactorial * counter;
    END;

{return the answer to the user}
  writeln(number : 2, ' factorial is: ', theFactorial : 6 : 2);

END.
```

15.2 BASIC Version

I can not think of a personal computer that does not have at least one version of BASIC (Beginners All-purpose Symbolic Instruction Code) available for it. It was developed in the mid 60's by Thomas Kurtz and John Kemeny to be an easy-to-use language for beginners. In the 70's when programs came on cassette tape instead of floppy disc, BASIC was usually the only language available for use with your personal computer. You will find a plethora of books available for writing programs in BASIC in any decent bookstore.

```
10 REM this BASIC program will prompt the user for a positive integer
20 REM and it will then calculate and return the factorial of that number
30 REM i.e. given N, the program calculates N!
40 REM
45 REM get number from user
50 PRINT "Welcome to the factorial calculator"
60 INPUT "Please enter a positive integer number:", NUMBER
70 REM
80 REM calculate the factorial
90 THEFACTORIAL = 1
100 FOR COUNTER = 1 TO NUMBER
110 THEFACTORIAL = THEFACTORIAL * COUNTER
120 NEXT COUNTER
130 REM
140 REM return the answer to the user
150 PRINT NUMBER, " factorial is ", THEFACTORIAL
```

15.3 LISP version

LISP is a language used in Artificial Intelligence work. Its name is an acronym that stands for LISt Processing. LISP was developed in the mid 50's by John McCarthy, and is still the primary language used in artificial intelligence.

```
(defun factorial ()

; This Lisp program will prompt the user for a positive integer
; and it will then calculate and return the factorial of that number
; i.e. given N, the program calculates N!

  (print "Welcome to the factorial calculator:")
  (princ "Please enter a positive integer number:")
  (setf Number (read))

  ; calculate the factorial

  (setf TheFactorial 1)
  (do ((counter 1))
    ((> counter Number))
    (setf TheFactorial (* TheFactorial Counter))
    (setf counter (1+ counter))
  )

  ; return the answer to the user

  (princ Number)
  (princ " factorial is ")
  (princ TheFactorial)
  (terpri)
)
```

15.4 C version

C was developed in the early 70's by Dennis Ritchie to be part of the UNIX operating system. As a result many of the features in C allow flexible interfacing with UNIX. Originally UNIX was written in PDP-11 assembly code (very low level computer language) and occupied approximately 64K. The C version of UNIX turned out to be only 10% larger and hence C has now been adopted for developing operating systems and programming languages.

```
main()
{
  /* This C program will prompt the user for a positive integer */
  /* and it will then calculate and return the factorial of that number */
  /* i.e. given N, the program calculates N! */

  int number;            /* user input number */
  int counter;
  float theFactorial;    /* factorial of user number */

  /* Get number from user */
  printf("Welcome to the factorial calculator\n");
  printf("Please enter a positive integer number:");
  scanf("%d",\&number);

  /* Calculate the factorial */
  theFactorial = 1;

  for (counter = 1; counter <= number; counter++)
    theFactorial *= counter;

  /* Return the answer to the user */
  printf("%d factorial is: %f\n",number,theFactorial);
}
```

15.5 Compiling vs Interpreting

You may hear Pascal referred to as a compiled language. Running your program you may see the word "compiling" appear on the screen. Compiling is the act of translating the computer program into assembly language, so that the computer can run it. Languages such as Pascal, and C are compiled languages. Some other languages such as BASIC are interpreted languages. So now we need to explain the difference between a compiler and an interpreter.

Imagine you have this really neat book written in a foreign language, and you want to read it. Now you have a friend who can translate the book for you. The translator can sit down with the book, and you, and possibly some dictionaries and other helpful reference books. He can read one sentence out of the book, then tell you what that sentence said. This pattern continues until the end of the book. Now what happens if you need the book read to you again a few weeks later. You call up your friend but he isn't available. You are stuck. What you could have done when you had your translator friend available, was to ask him to REWRITE the book in English for you. He would go through the book sentence by sentence, but instead of reading you the sentence, he would write it into a new book. When he is finished you will have an English version of the book, and no need to call your friend again. This translation would have taken longer to do since the translator had to rewrite the book, but now that you have the translated copy, you can read it much quicker than before.

When your friend reads line by line and tells you what it says, he is interpreting the book for you. When he reads line by line and writes out a new version that you can understand, he is compiling a new version of the book for you. An interpreted computer program always needs the interpreter around to translate it so the computer can understand. A compiled computer program does not need an interpreter. The program has been compiled into a form that the computer understands.

If you have a program written in Pascal, you can interpret it from within any Pascal. Alternatively, you can Compile the program and save the compiled version onto a disc. This compiled version can be run separately from Pascal, and it runs much faster than the interpreted version. The Pascal program is portable. It can be moved onto different machines. On the other hand, the compiled (translated) version is not portable, and can only run on machines of exactly the same type. The compiled code is not as portable as the uncompiled code.

Interpreting is good when you are developing a program. When you have the program working the way you want it to, then you compile it. When you compile a

Pascal program and save it as an application, that application will only work on one type of machine, where the Pascal program can run on several different machines. I can type the factorial program into Turbo Pascal on an IBM-PC. I can type that same program into Lightspeed Pascal on the Mac. Once I compile the program and save it, however, it is in a form that is very specific to the machine that it was compiled on. This is one of the reasons why the compiled version runs faster.

15.6 Questions

1. Why do we want to compile our programs?

Chapter 16

Neat Stuff

Now that you have seen what you can do with a computer, the computer has lost a lot of its mystique. We couldn't let you go thinking that computers are really just as boring as toasters.

Understanding how our technology works helps us to understand ourselves. One of the great mysteries of ourselves is how the brain works, and the simple but hard question "What is intelligence?" It is a simple question to ask, but a very hard question to answer. Huge sums of money are spent to try to make computers act in intelligent ways. The trouble in doing this is that the problem is not well defined. We can not say how the brain works, so how can we create a device that works in a similar manner.

We do know that there are some things the brain does very well that computers are very bad at such as vision, and natural language understanding. Both of these are problems for computers because they involve a lot of noise. Human beings are very good at filtering out what is unimportant, and making decisions based on incomplete information. Computers are not. For example: When you are at a party talking with your friends there may be many other conversations going on in that same room; yet you can pick out what your friends are saying from all of the other words being spoken. Another example: When you are driving a car at night in a thunderstorm there is rain on the windshield, the wipers are swishing back and forth, car lights are reflecting off the water, yet you are able to keep the car on the road.

Two major methods have been proposed to make machines "intelligent." One is Artificial Intelligence, and the other is Neural Networks. At various times during the last 40 years both of these methods have been hailed as the be all and end all of computing. Of course neither has lived up to its hype, but each represents a different

way of thinking about intelligence.

16.1 Artificial Intelligence

Most "intelligent" systems developed so far have been Artificial Intelligence systems. AI typically uses rules to guide the computer's actions. If situation X then do action Y. This works very well for situations where we can supply rules to the computer. Many commercial AI systems are called Expert Systems. In this case an expert (or maybe a handful of experts) has told the computer all the rules that are used in a given situation. The computer can store all these rules and quickly access them. As the computer makes mistakes it can modify these rules. In effect, the computer learns.

These systems have had good success in areas such as medicine and chess. In both cases, given a certain situation a certain action should be performed. In medicine a certain set of symptoms may lead the computer to ask for specific tests to be run to isolate the problem. In chess a certain layout of pieces will lead the computer to move a specific piece. In both these areas computers are coming near the level of human practitioners.

Computerized game players have always been a popular area of study. Imagine a very simple game such as tic-tac-toe. Back when you were very young you could play this game endlessly with a friend. Then one day you realized that if you started in the center, you won much more often. In fact, it was almost impossible to lose if you started in the center.Within a week you gave up the game. You figured out the basic rules of winning the game. You could then move onto harder games. AI people have created successful computerized tic-tac-toe systems and checker players. In chess, computers are getting very close to knocking off the top human player. Of course then there are always harder games such as 'shogi' or 'go' to move onto. These kinds of programs have become so popular that most personal computers have a chess program available where the computer will play you.

One of the problems with artificial intelligence is that there are times when there is not a definite set of rules. Given a problem, human beings do not tend to follow a strict set of rules. Commonly we use "rules of thumb" to solve our problems. They do not always guarantee success, but they seem to work pretty well. Think about driving a stick-shift. You can't really explain how you know when to shift gears, you just know when to do it. It's a combination of vision, hearing, and feeling that you can't explain to someone who only drives an automatic. In computer jargon these

"rules of thumb" are called heuristics. It is difficult to express these heruistics in the form of rules.

16.2 Neural Networks

Neural Networks take a different approach, which some consider to be much closer to the way the human brain works. Instead of having a list of rules, neural networks have layers of cells which are stimulated by other cells, much as the brain's neurons are stimulated. As the system "learns" the connections between the cells are altered. These systems are good at filtering out noise, and matching patterns, but they are not very good at math.

The HAL 9000 computer from "2001" was supposedly an advanced neural network computer (Heuristically program ALgorithmic computer.) By the way if you shift each letter in HAL, to the right one letter, you get IBM!

16.3 Intelligence

Being intelligent is not just being able to recall previous information that you learned. Given a new situation an intelligent system refers back to similar previous experiences to help make decisions about what do do in the present situation. But how do you store those past experiences? How do you know what are the most important parts of your past experiences? How do you relate your current experiences to your past experiences? There are a lot of questions. With the artificial intelligence system one can look at the rules and see how the computer has modified them. We can see what it has learned. It is difficult to tell how neural networks are solving a problem, so it is harder to gauge their progress.

Back in 1950 Alan Turing proposed the 'Turing Test' for determining whether a machine was intelligent. The test works like this. You stand in front of a locked door, and you do not know who or what is behind the door. You can communicate with the entity behind the door (perhaps by typing on a keyboard, or writing on a piece of paper or talking) and have a conversation. Let's say there is a computer behind the door. If you can not tell that it is a computer and not a human being then that computer has passed the Turing test, and must be considered 'intelligent.'

In 1980 John Searle discussed what he (and many others) consider a flaw in the 'Turing Test.' Searle proposed a similar situation to the Turing Test and called it

the 'Chinese Room.' Let's say we again have a locked door and the person outside the door will communicate with the person behind the door in written Chinese - that is, the person outside the door will write chinese characters onto a piece of paper, pass that piece of paper under the door, and after a certain amount of time receive another sheet of paper with Chinese characters written on it in reply. John Searle put himself behind the locked door. Now Mr. Searle does not know how to read or write Chinese, but he takes with him a large set of rules written in English which tell him that whenever he receives a certain set of Chinese characters he should write the following characters on his sheet of paper in reply and pass that sheet back out the door. Mr. Searle can now have a meaningful discussion in Chinese with the person outside of the door without ever understanding a word of what he is writing. Searle implies that we can program a machine to mimic human behavior without human understanding.

16.4 The Future

Criswell said it best in the opening of *Plan 9 from Outer Space*

> We are all interested in the future for that is where you and I are going to spend the rest of our lives.

There is still a lot of room for progress in these systems. No computer system will ever be the answer to all our problems. The computer is just another drip in a stream of ever more sophisticated tools. Throughout time we have created machines to help us solve our physical problems ... machines to move the earth, or to help us fly over it. Now we create machines to help us understand our world and ourselves. As with all our machines, they show us who we are, and what we are striving to become.

With all of our creations there is the promise and the peril. Computers give us even more opportunities, but we need to decide how to use this new freedom. Will it help bring us together, or tear us apart? Will it help expand our horizons, or close our minds? Will it lead us into a new age, or will it push us back?
It can do none of these things.

Only we can.

We decide.

Appendix A

Other Books

If you wish to read further on some of the subjects we have talked about in this text, you may find the following books to be of interest:

General Information:

- Ditlea,Steve *Digital Deli* Workman Pub. 1984
- Kiser, Denise *Computing Unbound* W.W. Norton & Co. 1989
- Levy, Steven. *Hackers - Heroes of the Computer Revolution* Dell Pub. 1984.
- Penzias, Arno *Ideas and Information - Managing in a High-Tech World* W.W. Norton & Co. 1989.

Operating Systems:

- Waite,Mitchell *UNIX Primer Plus* Howard W Sams & Co. Inc.

Word Processing:

- Erickson,Tim *Desktop Publishing with Microsoft Word on the Macintosh* SYBEX inc.

Telecommunications:

- Hedtke,John V *Using Computer Bulletin Boards* MIS: Press. 1990

- Stoll, Clifford *The Cuckoo's Egg* Double-day. 1989

Databases:

- Date, C.J.*Introduction to Database Systems.* Addison-Wesley. 1987
- Korth, Henry *Database System Concepts* McGraw-Hill. 1986.

Hypertext:

- Goodman, Danny.*The Complete HyperCard Handbook.* Bantam. 1987
- Vaughan, Tay. *Using Hypercard, from Home to Hypertalk.* Que Corp.

Programming:

- Aho, Alfred *Data Structures and Algorithms* Addison-Wesley. 1985.
- Koffman, Elliot *Problem Solving and Structured Programming in Pascal.* Addison-Wesley. 1986
- Winston, Patrick *Lisp* Addison Wesley. 1984.

Neat Stuff:

- Gardner, Howard.*The Mind's New Science.* Basic Books. 1987.
- Winston, Patrick *Artificial Intelligence* Addison-Wesley. 1984.

Appendix B

Other Periodicals

For truly up to date information there are several periodicals that cover the personal computer area. With the increasing popularity of personal computers, many of these are available at your local library.

Info World	Weekly	$2.95	Business software/hardware	12th yr.
ComputerWorld	Weekly	$2.00	Business software/hardware	24th yr.
Byte	Monthly	$3.50	General software/hardware	15th yr.
Personal Computing	Monthly	$3.00	Home software/hardware	14th yr.
AI expert	Monthly	$3.50	Information on AI for novices	5th yr.
Computer Language	Monthly	$3.50	Computer Languages	7th yr.
Mac World	Monthly	$3.95	Mac software/hardware	7th yr.
Mac User	Monthly	$2.95	Mac software/hardware	6th yr.
Amiga World	Monthly		Amiga related software/hardware	
PC World	Monthly	$2.95	IBM-pc software/hardware	8th yr.
PC Magazine	Monthly	$2.95	IBM-pc software/hardware	9th yr.
Incider/A+	Monthly	$3.95	Apple][software/hardware	

Appendix C

Professional Societies

Anyone who is planning on making a career in computer science should belong to at least one of the major professional organizations. These organizations publish periodicals on a wide range of current topics, and hold conferences all over the world. They have active local chapters at most major universities, and best of all they offer substantial discounts to student members.

On the engineering side (Electronics Engineer, Computer Engineer) there is the IEEE, and specifically its computer society.

```
IEEE

The Institute of Electrical and Electronics Engineers
Founded in 1963
300,000 current members
main publication: IEEE SPECTRUM

    IEEE Computer Society
    Founded in 1963
    100,000 current members
    main publication: COMPUTER
```

On the Computer Science side there is the ACM.

```
ACM
```

```
The Association for Computing Machinery
Founded in 1947
80,000 current members
main publication: COMMUNICATIONS OF THE acm
```

Appendix D

Acronyms

You have probably noticed that there are a lot of acronyms in this book. Here's a list of some of the more common ones in computer science:

ACM	Association for Computing Machinery
AI	Artificial Intelligence
ALGOL language	Algorithmic Oriented Language
ALU	Arithmetic Logic Unit
ANSI	American National Standards Institute
APL language	A Programming Language
ASCII	American Standard Code for Information Interchange
ATM	Automated Teller Machine
BAR	see FOOBAR
BASIC language	Beginners All-purpose Symbolic Instruction Code
BBS	Bulletin Board System
BIT	Binary digIT
CAD/CAM	Computer Aided Design/ Computer Aided Manufacturing
CAEN	Computer Aided ENgineering
CD	Compact Disc
CIS	Compuserve Information System
CLI	Command Line Interface
COBOL language	COmmon Business Oriented Language
CPU	Central Processing Unit
CRT	Cathode-Ray Tube
DB	DataBase

DBMS	DataBase Management System
DIP chip	Dual In-line Package
DIY	Do It Yourself
DOS	Disc Operating System
DPI	Dots Per Inch
EBCDIC	Extended Binary Coded Decimal Interchange Code
FOO	see FOOBAR
FOOBAR (FUBAR)	F*ck*d Up Beyond All Recognition
FORTRAN language	FORmula TRANslator
FBI	Federal Bureau of Investigation
FCC	Federal Communications Commission
GUI	Graphical User Interface
IBM	International Business Machines
IEEE	Institute of Electrical and Electronics Engineers
LAN	Local Area Network
LD	Laser Disc
LISP language	LISt Processing
MIS	Management Information Systems
MODEM	MOdulator DEModulator
I/O	Input/Output
OS	Operating System
PC	(IBM) Personal Computer
PET computer	Personal Electronic Transactor
PL/I language	Programming Language I
RAM	Random Access Memory
RGB monitor	Red Green Blue monitor
ROM	Read Only Memory
RPM	Revolutions Per Minute
SCSI	Small Computer System Interface ("skuzzy")
SNAFU	Situation Normal - All F*ck*d Up
SQL	Semantic Query Language (" sequel")
SYSOP	SYStem OPerator
TRS	Tandy Radio Shack
TV	TeleVision
VCR	Video Casette Recorder
VDT	Video Display Terminal

WIMP	Windows Icons Menus and a Pointer
WYSIWYG	What You See Is What You Get ("wizzywig")

Appendix E

Glossary

Algorithm A step by step procedure for solving a problem in finite time, also known as a recipe.

Application A computer program that performs a specific task such as a Word Processor or a Database.

Baud Rate The speed at which modems communicate. Typical speeds are 1200 baud or 2400 baud.

BBS A computerized Bulletin Board System where users leave messages for each other.

Bit A Binary digIT (ie a single 0 or 1).

Bug An error in your program.

Byte A string of 8 bits (eg. 11001001).

Card A small printed circuit board that is attached to the motherboard with a card-edge connector. It is attached to expand the capabilities of the computer.

Chip A DIP Chip contains a small integrated cicrcuit that processes information in the form of electrical signals.

Clone A computer made by company X that runs just like a machine built by company Y.

Command Line Interface A way of communicating instructions to a computer where the user types commands at a prompt.

Comments The part of a computer program that was written for humans to read, which the computer ignores.

Compiler A program that translates another program from a form that people can understand to a form a computer can understand.

Computer A programmable electronic device which stores, processes and retrieves data.

CPU The Central Processing Unit is the most important chip in the computer which does most of the processing.

Database An application used to manage large amounts of data where quick retrieval and update is necessary.

Desktop Publishing The use of a personal computer to prduce professional looking documents.

Directory A grouping of related files within a hierarchical file structure that is pictorially represented by a folder.

Disc A secondary storage medium which uses a thin round disc with a magnetic coating to store information. These discs can either be Floppy Discs used in Floppy Disc Drives or Hard Discs used in Hard Disc Drives.

Document processor An application used in conjuction with a text editor to typeset a large document according to a specific style.

DOS The Disc Operating System which handles all the low level interaction with the disc drives.

E-Mail Messages sent via computer.

Emulator Hardware and/or software that allows one brand of computer to run software designed for a different brand of computer.

Ergonomics The science of designing equipment so it will be easy to use by people.

File Collection of related data stored on a disc.

Floppy Drive A disc drive which can store small (1 M) amounts of information on a small floppy disc which can then be taken with the user wherever he wishes to take it.

Fone Phreak Someone who is interested in telecommunications systems, usually so they can make free phone calls.

Formatting Preparing a new disk for storing information.

Graphical User Interface A way of communicating instructions to a computer where the user selects from options displayed on the screen using a mouse or other device which gives her a surrogate hand on the monitor screen.

Hacker (modern definition) Someone who illegaly enters other people's computer systems.

Hard Drive A disc drive which can store and retrieve large (50 M) amounts of information quickly due to the magnetic disc not being removable.

Hardware The physical parts of the computer including the monitor, the main unit and the printer.

Hierarchical File Structure A way of organizing files on a disc so that related objects are together within a hierarchy of different subject areas.

High Level Language An 'English-like' programming language.

Hypermedia The linking of a hypertext program to multiple media such as LD players, or CD players to allow the program to manage visual and audio information.

Hypertext A system which links information together allowing a user to move quickly between important pieces of related information.

Icon The pictorial representation of an object or idea.

Instruction The smallest unit of control for a computer. A complex program is made up of many simple instructions.

Interface the means by which you communicate with the computer. This usually includes the monitor, the keyboard, and a mouse.

K A string of 1024 bytes.

Key a) The metal device which allows you to enter your apartment or b) a column (or several columns) that allows you to find a unique row in a database table.

Keyboard The primary input device for a computer which looks very much like the keys on a typewriter.

Laptop A small, portable personal computer that use can set on your lap to use.

Literate Programming A style of writing programs where the program is readable to non computer-scientists.

Low Level Language A 'computer like' programming language which is easily understood by the computer, but not by humans.

M A string of 1024 K.

Main Memory see RAM.

Main Unit The big box that contains the guts of the computer. It is usually attached to the monitor and the keyboard by cables.

Modem A device which allows computers to send information over standard phone lines by converting the information into sound.

Monitor The thing that looks like a TV set, and allows you to get output from the computer in a visual manner.

Motherboard The main printed circuit board in the computer where all the important chips are located.

Mouse A hand held input device good for making selections, and doing crude drawings.

Multitasking The ability of a computer to run more than one application at the same time.

OS The computers Operating System which handles all the low level work that the computer does.

Password A secret, personal identification code.

Pirate a) Errol Flynn in the movie "Captain Blood" or b)Someone who illegally copies software.

Pixel An individual 'dot' on the screen.

Printed Circuit Board A fiberglass or epoxy sheet which acts as a mounting board for various electronic components. Cards and the Motherboard are printed circuit boards.

Printer A device used to create a printed version of information stored in the computer.

Program A sequence of instructions detailing the steps to be performed by a computer.

Programming language A language used to write computer programs.

Protocol A common language that modems must agree on to allow different brands of computers to communicate via phone lines.

Pseudo-Code Informal algorithmic notation mixing English and code.

Public Domain Software Software that does not cost any money which is free for you to copy and distribute.

Query Language The language in which a person communicates with a database.

RAM The volatile Random Access Memory in a computer.

Recursion See: Recursion.

ROM The involatile Read Only Memory in a computer.

Secondary Storage Permanent storage usually on discs or tape.

Software Programs that tell the hardware what to do.

Source-Code A program in its original form, in the programming language it was written, before being compiled.

Speaker A device which converts electrical signals into sounds.

Spreadsheet An application which allows you to do mathematical calculations on a screen which looks like an accountant's ledger pad.

Stepwise Refinement The process of breaking a task down into smaller more specific sub tasks.

Symantic Error An error in meaning. You have instructed the computer to carry out a specific action and the machine has done this, but it is not the correct action.

Syntax Error An error in grammer where the machine does not understand what you want it to do.

Sysop The System Operator, and manager of a BBS.

Terminal Program An application that allows your computer to talk with other computers using a modem.

Text editor A no frills word processor used mainly for writing computer programs and producing input for document processors.

Trojan Horse An application that does one thing while pretending to do something else.

User Group A group of users of a certain brand of computer who get together to discuss new pieces of hardware and software, frequently run by xenophobic little führers who believe that their chosen computer is the best computer for everyone.

Virus A small program designed to hide itself within other programs and spread to as many other programs on as many other discs as possible. Some are benign and some can cause great hassles, but the media has blown them way out of proportion.

Word processor An application which turns your computer into a very sophisticated typewriter with lots of optional goodies.