Title: Chapter 1 Introduction to Computers and Programming

Topic:

1.1 Why Program?

1.2 Computer Systems: Hardware and Software

1.3 Programs and Programming Languages

1.4 What is a Program Made of?

1.5 Input, Processing, and Output

1.6 The Programming Process

1.7 Tying It All Together: Hi! It’s Me

# 1.1 WHY PROGRAM?

Concept: Computers can do many different jobs because they are programmable.

The computer is a tool that is used by so many professions, it cannot be easily categorized. It can perform so many different jobs that it is perhaps the most versatile tool ever made.

Q:

What makes computers so useful?

A:

Quite simply, the computer can do such a wide variety of tasks because it can be programmed. It is a machine specifically designed to follow instructions.

Computer programmers do a very important job. They create software that transforms computers into the specialized tools of many trades. Without programmers, the users of computers would have no software, and without software, computers would not be able to do anything.

Computer Programming is both an art and a science. It is an art because every aspect of a program should be designed with care and judgement.

List of things that must be designed for any real- world computer program:

* The logical flow of the instructions
* The mathematical procedures
* The appearance of the screens
* The way information is presented to the user
* The program’s “user-friendliness”
* Manuals and other forms of written documentation

There is also a scientific or engineering side to programming. Because programs rarely work right the first time they are written, a lot of experimentation, correction, and redesigning is required. This demands patience and persistence of the programmer.

Writing software demands discipline as well.

Programmers must learn special languages like C++ because computers do not understand English or other human languages. Languages such as C++ have strict rules that must be carefully followed.

# 1.2 Computer Systems: Hardware and Software

Concept: All computer systems consist of similar hardware and software components. This section provides an overview of standard computer hardware and software organization.

Hardware

Hardware refers to the physical components that a computer is made of. A computer, as we generally think of it, is not an individual device, but a system of devices. A typical computer system consists of the following major components:

* The central processing unit (CPU)
* Main memory (random-access memory, or RAM)
* secondary storage devices
* Input devices
* Output devices

The CPU

At the heart of a computer is the central processing unit, or CPU. The CPU’s job is to fetch instructions, follow the instructions, and produce some result. Internally, the central processing unit consists of two parts: the control unit and the arithmetic and logic unit (ALU). The control unit coordinates all of the computer’s operations. It is responsible for determining where to get the next instruction and regulating the other major components of the computer with control signals. The arithmetic and logic unit, as its name suggests, is designed to perform mathematical operations.

A program is a sequence of instructions stored in the computer’s memory. When a computer is running a program, the CPU is engaged in a process known formally as the fetch/decode/execute cycle. The steps in the fetch/decode/execute cycle are as follows:

Fetch The CPU’s control unit fetches, from main memory, the next instruction in the sequence of program instructions.

Decode The instruction is encoded in the form of a number. The control unit decodes the instruction and generates an electronic signal.

Execute The signal is routed to the appropriate component of the computer (such as the ALU, a disk drive, or some other device). The signal causes the component to perform an operation.

* These steps are repeated as long as there are instructions to perform.

Main Memory

Commonly known as random-access memory, or RAM, the computer’s main memory is a device that holds information. Specifically, RAM holds the sequence of instructions in the programs that are running and the data those programs are using.

Memory is divided into sections, or cells, that hold an equal amount of data, each cell typically contains eight “switches” that may be either on or off. A switch that is in the on position usually represents the number 1, while a switch in the off position usually represents the number 0. The computer stores data by setting the switches in a memory sell to a pattern that represents a piece of information. Each of these switches is known as a bit, which stands for binary digit. Each cell, which is a collection of eight bits, is known as a byte.

Bytes are grouped together to make words. On most computers a word contains four bytes. Each word is assigned a unique number known as an address. The addresses are ordered from lowest to highest. A word is identified by its address in much the same way a post office box is identified by an address.

RAM is usually a volatile type of memory, used only for temporary storage. When the computer is turned off, the contents of RAM are erased.

Secondary Storage

Secondary storage is a type of memory that can hold data for long periods of time—even when there is no power to the computer. Frequently used programs are stored in secondary memory and loaded into main memory as needed. Important information, such as word processing documents, payroll data, and inventory figures, is saved to secondary storage as well.

The most common type of secondary storage device is the disk drive. a disk drive stores data by magnetically encoding it onto a circular disk. Most computers have a disk drive mounted inside their case. External disk drives, which connect to one of the computers communication ports, are also available. External disk drives can be used to create backup copies of important data or to move data to another computer.

In addition to external disk drives, many types of devices have been created for copying data and for moving it to other computers. The use of floppy disk drives has declined dramatically in recent years, in favor of superior devices such as USB flash drives. USB flash drives are small devices that plug into the computer’s USB (universal serial bus) port and appear to the system as a disk drive. These drives, which use flash memory to store data, are inexpensive, reliable, and small enough to be carried in your pocket.

Optical devices such as the CD (compact disc) and the DVD (digital versatile disc) are also popular for data storage. Data is not recorded magnetically on an optical disc, but rather is encoded as a series of pits on the disc surface. CD and DVD drives use a laser to detect the pits and thus read the encoded data. Optical discs hold large amounts of data, and because recordable CD and DVD drives are now commonplace, they are good media for creating backup copies of data.

Input Devices

Input is any information the computer collects from the outside world. The device that collects the information and sends it to the computer is called an input device. Common input devices are the keyboard, mouse, scanner, digital camera, and microphone. Disk drives, CD/DVD drives, and USB flash drives can also be considered input devices because programs and information are retrieved from them and loaded into the computer’s memory.

Output Devices

Output is any information the computer sends to the outside world. it might be a sales report, a list of names, or a graphic image. The information is sent to an output device, which formats and presents it. Common output devices are computer screens, printers, and speakers. Output sent to a computer screen is sometimes called a soft copy, while output sent to a printer is called hard copy. Disk drives, USB flash drives, and CD/DVD recorders can also be considered output devices because the CPU sends information to them so it can be saved.

Software

As previously mentioned, software refers to the programs that run on a computer. There are two general categories of software: operating systems and application software. An operating system is a set of programs that manages the computer’s hardware devices and controls their processes. Operating systems fall into one of the following categories.

Single tasking A single tasking operating system is capable of running only one program at a time. The computer devotes all its hardware resources and CPU time to each program as it executes. MS-DOS is an example of a single tasking operating system.

Multitasking A multitasking operating system is capable of running multiple programs at once. Through a technique called time sharing, the system divides the allocation of hardware resources and the attention of the CPU among all the executing programs. UNIX, Window XP, and Windows Vista are multitasking operating systems.

In addition, operating systems fall into one of the following categories, which describe the number of users they can accommodate.

Single user This type of system allows only one user to operate the computer at a time. MS-DOS and older versions of Windows are single user operating systems.

Multiuser Multiuser systems allow several users to run programsadd operate the computer at once. Most variations of the UNIX operating system are multiuser systems.

Application software refers to programs that make the computer useful to the user. These programs solve specific problems or perform general operations that satisfy the needs of the user. Word processing, spreadsheet, and database programs are all examples of application software.

Checkpoint

1.1 Why is the computer used by so many different people, in so many different professions?

1.2 List the five major hardware components of a computer system.

1.3 Internally, the CPU consists of what two units?

1.4 Describe the steps in the fetch/decode/execute cycle.

1.5 What is a memory address?

1.6 Explain why computers have both main memory and secondary storage.

1.7 What are two general categories of software?

1.8 What is the difference between a single tasking system and a multitasking system?

1.9 What is the difference between a single user system and a multiuser system?

1.3 Programs and Programming Languages

Concept: A program is a set of instructions a computer follows in order to perform a task. A programming language is a special language used to write computer programs.

What is a Program?

Computers are designed to follow instructions. A computer program is a set of instructions that tell the computer how to solve a problem or perform a task. For example, suppose we want the computer to calculate someone’s gross pay. Here is a list of things the computer might do:

* Display a message on screen asking “How many hours did you work?”
* Wait for the user to enter the number of hours worked. Once the user enters a number, store it in memory.
* Display a message on screen asking “How much do you get paid per hour?”
* Wait for the user to enter an hourly pay rate. Once the user enters a number, store it in memory
* Multiply the number of hours by the amount paid per hour, and store the result in memory.
* Display a message on the screen that tells the amount of money earned. The message must include the result of the calculation performed in step 5.

Collectively, these instructions are called an algorithm. An algorithm is a set of well defined steps for performing a task or solving a problem. Notice these steps are sequentially ordered. Step 1 should be performed before step 2, and so forth. It is important that these instructions be performed in their sequence.

Although a person might easily understand the instructions in the pay-calculating algorithm, it is not ready to be executed on a computer. A computer’s CPU can only process instructions that are written in machine language. A machine language program consists of a sequence of binary numbers (numbers consisting of only 1s and 0s) which the CPU interprets as commands. Here is an example of what a machine language instruction might look like:

1011010000000101

As you can imagine, the process of encoding an algorithm in machine language is very tedious and difficult. In addition, each different type of CPU has its own machine language. If you wrote a machine language programmer for computer A and then wanted to run it on a computer B that has a different type of CPU, you would have to rewrite the program in computer B’s machine language.

Programming languages, which use words instead of numbers, were invented to ease the task of programming. A program can be written in a programing language such as C++, which is much easier to understand than machine language. Programmers save their programs i text files, and then use special software to convert their programs to machine language.

Program 1-1

// This program calculates the user’s pay

#include <iostream>

using namespace std;

int main()

{

double hours, rate, pay;

// Get the number of hours worked.

cout << “How many hours did you work? “;

cin >> hours;

// Get the hourly pay rate

cout << “How much do you get paid per hour? “;

cin >> rate;

// Calculate the pay

pay = hours \* rate;

// Display the pay.

cout << “You have earned $” << pay << endl;

return 0;

}

Programming Languages

In a broad sense, there are two categories of programming languages: low-level and high-level. A low-level language is close to the level of the computer, which means it resembles the numeric machine language of the computer more than the natural language of humans. The easiest languages for people to learn are high-level languages. They are called “high-level” because they are closer to the level of human-readability than computer-readability.

well-known high-level programming languages

Language Description

BASIC Beginners All-purpose Symbolic Instruction Code. A general programming language originally designed to be simple enough for beginners to learn.

C A structured, general-purpose language developed at Bell Laboratories. C offers both high-level and low-level features.

C++ Based on the C language, C++ offers object-oriented features not found in C. Also invented at Bell Laboratories.

C# Pronounced “C sharp.” A language invented by Microsoft for developing applications based on the Microsoft .NET platform.

COBOL Common Business-Oriented Language. A language designed for business applications.

FORTRAN Formula Translator. A language designed for programming complex mathematical algorithms.

Java An object-oriented language invented at Sun Microsystems. Java may be used to develop programs that run one the Internet in a Web browser.

JavaScript A language used to write small programs that run in Web pages. Despite its name, JavaScript is not related to Java.

Pascal A structured, general-purpose language designed primarily for teaching programming.

Python A general purpose language created in the early 1990s. It has become popular for both business and academic applications.

Ruby A general purpose language created in the early 1990s. It is becoming increasingly popular for programs that run on Web servers.

Visual Basic A Microsoft programming language and software development environment that allows programmers to quickly create Windows-based applications.

C++ is a widely used language because, in addition to the high-level features necessary for writing applications such as payroll systems and inventory programs, it also has many low-level features. C++ is based on the C language, which was invented for the purposes such as writing operating systems and compilers. Because C++ evolved from C, it carries all of C’s low-level capabilities with it.

C++ is also popular because of its portability. This means that a C++ program can be written on one type of computer and then run on many other types of systems. This usually requires that the program is recompiled on each type of system, but the program itself may need little or no change.

note: programs written for specific graphical environments often require significant changes when moved to a different type of system. examples of such graphical environments are windows, the x-window system, and the mac os x operating system.

Source Code, Object Code, and Executable Code

When a C++ program is written, it must be typed into the computer and saved to a file. A text editor, which is similar to a word processing program, is used for this task. The statements written by the programmer are called source code, and the file they are saved in is called the source file.

After the source code is saved to a file, the process of translating it to machine language can begin. During the first phase of this process, a program called the preprocessor reads the source code. The preprocessor searches for special lines that begin with the # symbol.These lines contain commands, or directives , that cause the preprocessor to amend or process the source code in some way.

During the next phase the compiler steps through the preprocessed source code, translating each source code instruction into the appropriate machine language instruction. This process will uncover any syntax errors that may be in the program.

Syntax errors are illegal uses or key words, operators, punctuation, and other language elements.

If the program is free of syntax errors, the compiler stores the translated machine language instructions, which are called object code, in an object file.

Although an object file contains machine language instructions, it is not a complete program.

Here is why.

C++ is conveniently equipped with a library of prewritten code for performing common operations or sometimes-difficult tasks.

For example, the library contains hardware-specific code for displaying messages on the screen and reading input from the keyboard.

It also provides routines for mathematical functions, such as calculating the square root of a number.

This collection of code, called the run-time library, is extensive.

Programs almost always use some part of it.

When the compiler generates an object file, however, it does not include machine code for any run-time library routines the programmer might have used.

Durning the last phase of the translation process, another program called the linker combines the object file with the necessary library routines.

Once the linker has finished with this step, or executable code, and is ready to run on the compiler.

Many development systems, particularly those on personal computers, have integrated development environments (IDE’s).

These environments consist of a text editor, compiler, debugger, and other utilities integrated into a package with a single set of menus.

Preprocessing, compiling, linking, and even executing a program is dine with a single click of a button, or by selecting a single item from a menu.

Checkpoint

1.10 What is an algorithm?

1.11 Why were computer programming languages invented?

1.12 What is the difference between a high-level language and a low-level language?

1.13 What does portability mean?

1.14 Explain the operations carried out by the preprocessor, compiler, and linker.

1.15 Explain what is stored in a source file, an object file, and an executable file.

1.16 What is an integrated development environment?

1.4 What is a Program Made of?

Concept: There are certain elements that are common to all programming languages.

Language Elements

All programming languages have a few things in common.

programming language elements

Language Element Description

Key Words Words that have a special meaning. Key words may only be used for their intended purpose. Key words are also known as reserved words.

Programmer-Defined Words or names defined by the programmer. Identifiers They are symbolic names that refer to variables or programming routines.

Operators Operators perform operations on one or more operands. An operand is usually a piece of data, like a number.

Punctuation Punctuation characters that mark the beginning or ending of a statement, or separate items in a list.

Syntax Rules that must be followed when constructing a program. Syntax dictates how key words and operators may be used, and where punctuation symbols must appear.

Key words (reserved words)

Three of C++'s key words: using, namespace, and int.

The word double is also a C++ key word.

These words, which are always written in lowercase, each have a special meaning in C++ and can only be used for their intended purposes.

Programmer-Defined Identifiers

They are not part of the C++ language but rather are names made up by the programmer.

Operators

The = and \* symbols are both operators.

They perform operations on pieces of data, known as operands. The \* operator multiples its two operands.

The = symbol is called the assignment operator.

It takes the value of the expression on the right and stores it in the variable whose name appears on the left.

Punctuation

Notice that many lines end with a semicolon.

A semicolon in C++ is similar to a period in English. It marks the end of a complete sentence (or statement, as it is called in programming).

Semicolons do not appear at the end of every line in C++ programs, however.

There are rules that govern where to place semicolons and other punctuation symbols.

Lines and Statements

Often, the contents of a program are thought of in terms of lines and statements.

A line is just that—a single line at is appears int the body of a program.

Blank lines are only there to make the program more readable.

A statement is a complete instruction that causes the computer to perform some action.

Statements can be a combination of key words, operators, and programmer-defined symbols.

Statements usually occupy only one line in a program, but sometimes they are spread out over more than one line.

Variables

A variable is a named storage location in the computer’s memory for holding a piece of data.

The data stored in variables may change while the program is running (hence the name “variable”).

Variables are symbolic names that represent locations in the computer’s random-access memory (RAM).

When information is stored in a variable, it is actually stored in RAM.

You can think of a variable as a box that holds information.

Only one item may be stored in the box at any given time.

Variable Definitions

In programming, there are two general types of data: numbers, such as 3, and characters, such as the letter ‘A’. numbers are used to perform mathematical operations and characters are used to print information on the screen or on paper.

When creating a variable in a C++ program, you must know what type of data the program will be storing in it.

In C++, all variable must be defined before they can be used because the variable definition is what causes the variables to be created in memory.

1.5 Input,Processing, and Output

Concept: The three primary activities of a program are input, processing, and output.

Computer programs typically perform a three-step process of gathering input, performing some process on the information gathered, and then producing output. Input is information a program collects from the outside world. It can be sent to the program by the user, who is entering data at the keyboard or using a mouse. It can also be read from disk files or hardware devices connected to the computer.

Once information is gathered from the outside world, a program usually processes it in some manner.

Output is information that a program sends to the outside world. It can be words or graphics displayed on the screen, a report sent to the printer, data stored in a file, or information sent to any device connected to the computer.

Checkpoint

1.17 Describe the difference between a key word and a programmer-defined symbol.

1.18 Describe the difference between operators and punctuation symbols.

1.19 Describe the difference between a program line and a statement.

1.20 Why are variables called “variable”?

1.21 What happens to a variable’s current contents when a new value is stored there?

1.22 What must take place in a program before a variable is used?

1.23 What are the three primary activities of a program?

1.6 The Programming Process

Concept: The programming process consists of several steps, which include design, creation, testing, and debugging activities.

Designing and Creating a Program

Now that you have been introduced to what a program is, it’s time to consider the process of creating a program. Quite often, when inexperienced students are given programming assignments, they have trouble getting started because they don't know what to do first.

If you find yourself in this dilemma, the steps listed may help. These are the steps recommended for the process of writing a program.

1. Define what the program is to do.
2. Visualize the program running on the computer.
3. Use design tools to create a model of the program.
4. Check the model for logical errors
5. Write the program source code.
6. Compile the source code.
7. Correct any errors found during execution.
8. Link the program to create an executable file.
9. Run the program using test data for input.
10. Correct any errors found while running the program.

Repeat steps 4 through 10 as many times as necessary.

11. Validate the results of the program.

What is Software Engineering?

The field of software engineering encompasses the complete process of crafting computer software. It includes designing, writing, testing, debugging, documenting, modifying, and maintaining complex software development projects. Like traditional engineers, software engineers use a number of tools in their craft. Here are a few examples:

* Program specifications
* Charts and diagrams of screen output
* Hierarchy charts
* Pseudocode
* Examples of expected input and desired output
* Special software designed for testing programs

Most commercial software applications are very large. In many instances one or more teams of programmers, not a single individual, develop them. It is important that the program requirements be thoroughly analyzed and divided into subtasks that are handled by individual teams or individuals within a team.

If the program is very large or complex, a team of software engineers can be assigned to work on the individual modules. As the project develops, the modules are coordinated to become a single software application.

Checkpoint

1.24 What four items should you identify when defining what a program is to do?

1.25 What does it mean to “visualize a program running”? What is the value of doing this?

1.26 What is a hierarchy chart?

1.27 What is pseudocode?

1.28 What is the difference between high level pseudocode and detailed pseudocode?

1.29 Describe what a compiler does with a program’s source code.

1.30 What is a logic error?

1.31 What is a run-time error?

1.32 Describe the process of desk-checking/

1.7 Tying It All Together: Hi! It’s Me

Most programs, as you have learned, have three primary activities: input, processing, and output. But it is possible to write a program that has only output.