Chapter 1

An Overview of Computers and Programming

• Overview

• Objectives

• Quick Quizzes

• Additional Resources

• Key Terms
Overview

Chapter 1 provides an introduction to computer hardware and to the software creation process. Students will be introduced to the basic steps involved in the programming process. They will learn about flowcharting and flowcharting symbols and become familiar with using sentinel values. Finally, students will learn about different types of programming and user environments.

Chapter Objectives

In this chapter, students will learn about:

- Computer systems
- Simple program logic
- The steps involved in the program development cycle
- Pseudocode statements and flowchart symbols
- Using a sentinel value to end a program
- Programming and user environments
- The evolution of programming models

Understanding Computer Systems

1. Review the major components of a computer system:
   a. **Hardware**
   b. **Software (programs)**

2. Know the difference between syntax and logical errors.

3. Review the example of the cake recipe on pages 5-6.

4. Review the simple example program to double a number on pages 6-7 and include a definition of the term variable.

Understanding the Program Development Life Cycle

1. List the seven phases of program development (Figure 1-1) before discussing each one in detail.
Understanding the Problem

1. Be able to point out that the failure to understand the problem to be solved is the major cause of software project failures. The most important sources for information about the problem are the end users and documentation.

The ability of a programmer to understand a user’s needs by observing the user’s job functions and questioning the user is a key skill. In many organizations, the step of understanding the problem is performed by application analysts, not programmers. In other organizations, programmers perform both analysis and programming and may hold the job title of programmer/analyst.

Planning the Logic

1. Explain that pseudocode allows a description of the program logic to be created without worrying about the details or syntax of a particular programming language.

The term “desk-checking” is also commonly used to describe the process of checking code for syntactical and logical correctness before compiling. This process was essential in earlier times when computing power was limited and scheduled and a programmer may have been able to complete only one compilation a day.

Coding the Program

1. Understand that the actual coding of the program does not occur until after the logic has been totally described and checked.

Using Software to Translate the Program into Machine Language

1. Describe the repetitive process of coding, compiling, and fixing syntax errors that must occur until the program is free of syntax errors. This is shown in Figure 1-2.

Testing the Program

1. Be able to point out the importance of selecting appropriate test data to ensure that all of the code is tested. Consider boundary conditions such as minimum and maximum values.

Putting the Program into Production

1. Explain that after testing, a program is put into production.

2. Define the term conversion, and explain that it may take months or years to complete.
Maintaining the Program

1. Describe the process of program maintenance.

Software testing, or software quality assurance, can be a career of its own. Today, automated testing tools are available to attempt a thorough review of all of the code in a program. The following article on how to choose test automation software can serve as a starting point for a discussion of test automation issues:
www.automatedqa.com/techpapers/selecting_automated_testing_tool.asp

An error-free test of the program does not guarantee that all errors have been corrected. The most one can say is that “all errors that have been identified have now been corrected.”

Quick Quiz 1

1. What are the two major components of a computer system?  
   Answer: hardware and software

2. What are the three basic operations of a computer?  
   Answer: input, processing, and output

3. What is the most important phase in creating a program to solve a problem?  
   Answer: understanding the problem

4. A computer program must be free of ____ errors before you can execute it.  
   Answer: syntax

Using Flowchart Symbols and Pseudocode Statements

1. Review the definitions of flowchart and pseudocode.

Writing Pseudocode

1. Walk through the example of pseudocode given on page 15 and understand the flexibility of pseudocode. Note that most programmers use start or begin to begin a program and end or stop at the end of the program.

Drawing Flowcharts

1. Describe each of the flowchart symbols and their purpose, using Figures 1-3 through 1-5 for illustration.
2. Describe flowlines and the depiction of direction with arrowheads on flowlines.
3. Describe the use of **terminal symbols**.

4. Use Figure 1-6 to understand the flowchart depiction of each line of pseudocode. This will help you understand how much detail should appear in pseudocode.

**Repeating Instructions**

1. Understand the need for repetitive processing of certain sets of program statements. This is achieved using a **loop**. Stress the importance of avoiding an **infinite loop**.

2. Describe how the use of directed flowlines indicates repetition in a flowchart using Figure 1-8.

   In Figure 1-8, show how the directed flowlines show the code that forms the loop. Point out the necessity for a decision somewhere inside the loop to avoid an infinite loop.

**Using a Sentinel Value to End a Program**

1. Describe an infinite loop and methods of stopping a loop during processing.

2. Describe **making a decision** in a program, and introduce the flowchart **decision symbol**. Ensure that students understand the need to document both the Yes and No paths in a decision symbol.

3. Understand the concept of a **sentinel** or **dummy value** to be used as a signal to stop the loop processing. Point out that if the loop is used to read files, the **end-of-file (eof)** marker is often used for this purpose.

4. Use Figures 1-9 and 1-10 to learn the use of a sentinel value in the decision point of the loop.

**Quick Quiz 2**

1. A symbol shaped like a(n) \__________ represents a decision in a flowchart. 
   Answer: diamond

2. A loop that runs forever is called a(n) \__________ loop. 
   Answer: infinite
3. What purpose does a sentinel value have?
   Answer: A sentinel value is used as a signal value to indicate that processing should stop.

4. A code stored in a file that marks the end of the data is called a(n) ____ marker.
   Answer: EOF or end-of-file

**Understanding Programming and User Environments**

1. Note that there are many different options when it comes to programming and user environments. These will be covered in the following sections.

**Understanding Programming Environments**

1. Describe the two main options for programming environments:
   a. Plain text editor (Figure 1-11)
   b. Text editor that is part of an integrated development environment (IDE) (Figure 1-12)

2. Review the features of an IDE listed on page 25.

**Understanding User Environments**

1. Understand the two main ways a user may interact with a program:
   a. Command line (Figure 1-13)
   b. Graphical user interface (GUI) (Figure 1-14)

**Understanding the Evolution of Programming Models**

1. Describe early programming, which required the use of direct memory addresses. Point out the high probability of errors, due to the fact that the code did not resemble a natural language.

   Note that low-level languages also required a much deeper understanding of how the computer worked internally, as there was close to a one-to-one relationship between the instruction and the machine action it caused.

2. Higher-level languages provide more than just ease of understanding the language. Many instructions in high-level languages perform several low-level operations automatically.

3. Programming design in early days tended to be monolithic, while newer programming design builds reusable modules of code.
4. Describe and contrast the two major programming techniques: **procedural programming** and **object-oriented programming**.

Many students think that OO techniques are unique in that they employ reusable code. Understand that the more important difference is the focus on objects as a programming technique.

**Quick Quiz 3**

1. (True/False) Many approaches can be used to write and execute a computer program.  
   Answer: True

2. A(n) ____ is a program that you use to create simple text files.  
   Answer: text editor

3. The ____ is a location on your computer screen at which you type text entries to communicate with the computer’s operating system.  
   Answer: command line

4. ____ programming focuses on objects, or “things,” and describes their features (or attributes) and their behaviors.  
   Answer: Object-oriented

**Additional Resources**

1. A short history of the computer:  

2. Article on software development process:  

3. Article on pseudocode:  

4. Article on IDEs:  

**Key Terms**

- **Algorithm** – the sequence of steps necessary to solve any problem.
- **Application software** – comprises all the programs you apply to a task.
- **Binary decision** – a yes-or-no decision with two possible outcomes.
- **Binary language** – represented using a series of 0s and 1s.
- **Central processing unit (CPU)** – the hardware component that processes data.
- **Coding the program** – the act of writing programming language instructions.
- **Command line** – a location on your computer screen at which you type text entries to communicate with the computer’s operating system.
- **Compiler/interpreter** – translates a high-level language into machine language and tells you if you have used a programming language incorrectly.
- **Computer memory** – the temporary, internal storage within a computer.
- **Computer system** – a combination of all the components required to process and store data using a computer.
- **Conversion** – the entire set of actions an organization must take to switch over to using a new program or set of programs.
- **Data items** – include all the text, numbers, and other information processed by a computer.
- **Debugging** – the process of finding and correcting program errors.
- **Decision symbol** – shaped like a diamond and used to represent decisions in flowcharts.
- **Desk-checking** – the process of walking through a program solution on paper.
- **Documentation** – consists of all the supporting paperwork for a program.
- **Dummy value** – a preselected value that stops the execution of a program.
- **eof** – means “end of file.”
- **Flowchart** – a pictorial representation of the logical steps it takes to solve a problem.
- **Flowlines** – or arrows, connect the steps in a flowchart.
- **Graphical user interface (GUI)** – allows users to interact with a program in a graphical environment.
- **Hardware** – the collection of physical devices that comprise a computer system.
- **High-level programming language** – supports English-like syntax.
- **Infinite loop** – occurs when repeating logic cannot end.
- **Input** – describes the entry of data items into computer memory using hardware devices such as keyboards and mice.
- **Input symbol** – indicates an input operation and is represented by a parallelogram in flowcharts.
- **Input/output symbol or I/O symbol** – represented by a parallelogram in flowcharts.
- **Integrated development environment (IDE)** – a software package that provides an editor, compiler, and other programming tools.
- **IPO chart** – a program development tool that delineates input, processing, and output tasks.
- **Logic** of a computer program – when you give instructions to the computer in a specific sequence, without omitting any instructions or adding extraneous instructions.
- **Logical error** – occurs when incorrect instructions are performed or when instructions are performed in the wrong order.
- **Loop** – a repetition of a series of steps.
- **Low-level language** – made up of 1s and 0s that the computer understands; also called machine language.
- **Machine language** – a computer’s on/off circuitry language.
- **Maintenance** – consists of all the improvements and corrections made to a program after it is in production.
- **Making a decision** – the act of testing a value.
- **Microsoft Visual Studio IDE** – a software package that contains useful tools for creating programs in Visual Basic, C++, and C#.
- **Nonvolatile** – describes storage whose contents are retained when power is lost.
- **Object code** – translated machine language.
- **Object-oriented programming** — a programming model that focuses on objects, or “things,” and describes their features (or attributes) and their behaviors.
- **Output** — describes the operation of retrieving information from memory and sending it to a device, such as a monitor or printer, so people can view, interpret, and work with the results.
- **Output symbol** — indicates an output operation and is represented by a parallelogram in flowcharts.
- **Procedural programming** — a programming model that focuses on the procedures that programmers create.
- **Processing** data items — may involve organizing them, checking them for accuracy, or performing mathematical operations on them.
- **Processing symbol** — indicates a processing operation and is represented by a rectangle in flowcharts.
- **Program code** — the set of instructions a programmer writes in a programming language.
- **Program development cycle** — consists of the steps that occur during a program’s lifetime.
- **Programming** — the act of developing and writing programs.
- **Programming languages** — Visual Basic, C#, C++, Java, or COBOL are used to write programs.
- **Programs** — sets of instructions for a computer.
- **Pseudocode** — an English-like representation of the logical steps it takes to solve a problem.
- **Random access memory (RAM)** — temporary, internal computer storage.
- **Run/execute** — carry out program’s instructions.
- **Scripting languages** (also called scripting programming languages or script languages) — such as Python, Lua, Perl, and PHP are used to write programs that are typed directly from a keyboard. Scripting languages are stored as text rather than as binary executable files.
- **Semantic error** — occurs when a correct word is used in an incorrect context.
- **Sentinel value** — a preselected value that stops the execution of a program.
- **Software** — consists of the programs that tell the computer what to do.
- **Source code** — the statements a programmer writes in a programming language.
- **Storage devices** — types of hardware equipment, such as disks, that hold information for later retrieval.
- **Syntax** — grammar rules of a language.
- **Syntax error** — an error in language or grammar.
- **System software** — comprises the programs that you use to manage your computer.
- **Terminal symbol** — or start/stop symbol, is used at each end of a flowchart. Its shape is a lozenge.
- **Text editor** — a program that you use to create simple text files; it is similar to a word processor, but without as many features.
- **TOE chart** — a program development tool that lists tasks, objects, and events.
- **Users (or end users)** — people who employ and benefit from computer programs.
- **Variable** — a named memory location whose value can vary.
- **Volatile** — describes storage whose contents are lost when power is lost.