Objectives

In this chapter, you 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

- **Computer system**
  - Combination of all the components required to process and store data using a computer

- **Hardware**
  - Equipment associated with a computer

- **Software**
  - Computer instructions
  - Tell the hardware what to do
  - **Programs**
    - Instructions written by programmers

Understanding Computer Systems (continued)

- **Programming**
  - Writing software instructions
  - Computer hardware and software accomplish three major operations
    - **Input**
      - Data items enter computer
    - **Processing**
      - By central processing unit (CPU)
    - **Output**

Understanding Computer Systems (continued)

- **Programming language**
  - Use to write computer instructions
  - Examples
    - Visual Basic, C#, C++, or Java

- **Syntax**
  - Rules governing its word usage and punctuation

- **Computer memory**
  - Computer’s temporary, internal storage
    - **Volatile**

Understanding Computer Systems (continued)

- **Permanent storage devices**
  - **Nonvolatile**

- **Compiler or an interpreter**
  - Translates program code into machine language (binary language)
  - Checks for syntax errors

- **Program executes or runs**
  - Input will be accepted, some processing will occur, and results will be output
Understanding Simple Program Logic

- Program with syntax errors cannot execute
- **Logical errors**
  - Errors in program logic
  - Produce incorrect output as a result
- Logic of the computer program
  - Sequence of specific instructions in specific order
- **Variable**
  - Named memory location whose value can vary

Understanding the Program Development Cycle

- **Program development cycle**
  - Understand the problem
  - Plan the logic
  - Code the program
  - Use software (a compiler or interpreter) to translate the program into machine language
  - Test the program
  - Put the program into production
  - Maintain the program

Understanding the Program Development Cycle (continued)

- **Figure 1-1 The program development cycle**

Understanding the Problem

- One of the most difficult aspects of programming
- **Users or end users**
  - People for whom program is written
- **Documentation**
  - Supporting paperwork for a program

Planning the Logic

- Heart of the programming process
- Most common planning tools
  - Flowcharts
  - Pseudocode
- **Desk-checking**
  - Walking through a program’s logic on paper before you actually write the program

Coding the Program

- Hundreds of programming languages are available
  - Choose based on features
  - Alike in their basic capabilities
  - Easier than planning step
Using Software to Translate the Program into Machine Language

- Translator program
  - Compiler or interpreter
  - Changes the programmer’s English-like high-level programming language into the low-level machine language
- Syntax error
  - Misuse of a language’s grammar rules
  - Programmer corrects listed syntax errors
  - Might need to recompile the code several times

Testing the Program

- Logical error
  - Use a syntactically correct statement but use the wrong one for the current context
- Test
  - Execute the program with some sample data to see whether the results are logically correct
- Programs should be tested with many sets of data

Putting the Program into Production

- Process depends on program’s purpose
  - May take several months
- Conversion
  - Entire set of actions an organization must take to switch over to using a new program or set of programs

Maintaining the Program

- Maintenance
  - Making changes after program is put into production
- Common first programming job
  - Maintaining previously written programs
- Make changes to existing programs
  - Repeat the development cycle

Using Pseudocode Statements and Flowchart Symbols

- Pseudocode
  - English-like representation of the logical steps it takes to solve a problem
- Flowchart
  - Pictorial representation of the logical steps it takes to solve a problem
Writing Pseudocode

• Pseudocode representation of a number-doubling problem
  ```
  start
  input myNumber
  set myAnswer = myNumber * 2
  output myAnswer
  stop
  ```

Writing Pseudocode (continued)

• Programmers preface their pseudocode with a beginning statement like `start` and end it with a terminating statement like `stop`
• Flexible because it is a planning tool

Drawing Flowcharts

• Create a flowchart
  – Draw geometric shapes that contain the individual statements
  – Connect shapes with arrows
• Input symbol
  – Indicates input operation
  – Parallelogram
• Processing symbol
  – Processing statements such as arithmetic
  – Rectangle

Drawing Flowcharts (continued)

• Output symbol
  – Represents output statements
  – Parallelogram
• Flowlines
  – Arrows that connect steps
• Terminal symbols
  – Start/stop symbols
  – Shaped like a racetrack
  – Also called lozenge

Drawing Flowcharts (continued)

![Figure 1-6 Flowchart and pseudocode of program that doubles a number](image)

Repeating Instructions

• After the flowchart or pseudocode has been developed, the programmer only needs to:
  – Buy a computer
  – Buy a language compiler
  – Learn a programming language
  – Code the program
  – Attempt to compile it
  – Fix the syntax errors
  – Compile it again
  – Test it with several sets of data
  – Put it into production
Repeating Instructions (continued)

• Loop
  – Repetition of a series of steps
• Infinite loop
  – Repeating flow of logic with no end

Figure 1-8 Flowchart of infinite number-doubling program

Using a Sentinel Value to End a Program

• Making a decision
  – Testing a value
  – Decision symbol
    • Diamond shape
• Dummy value
  – Data-entry value that the user will never need
  – Sentinel value
• eof ("end of file")
  – Marker at the end of a file that automatically acts as a sentinel

Figure 1-9 Flowchart of number-doubling program with sentinel value of 0

Figure 1-10 Flowchart using eof

Understanding Programming and User Environments

• Many options for programming and user environments
Understanding Programming Environments

- Use a keyboard to type program statements into an editor
  - Plain text editor
    - Similar to a word processor but without as many features
  - Text editor that is part of an integrated development environment (IDE)
    - Software package that provides an editor, compiler, and other programming tools

Understanding User Environments

- Command line
  - Location on your computer screen at which you type text entries to communicate with the computer's operating system
- Graphical user interface (GUI)
  - Allows users to interact with a program in a graphical environment

Understanding the Evolution of Programming Models

- People have been writing modern computer programs since the 1940s
- Newer programming languages
  - Look much more like natural language
  - Easier to use
  - Create self-contained modules or program segments that can be pieced together in a variety of ways
Understanding the Evolution of Programming Models (continued)

- Major models or paradigms used by programmers
  - **Procedural programming**
    - Focuses on the procedures that programmers create
  - **Object-oriented programming**
    - Focuses on objects, or “things,” and describes their features (or attributes) and their behaviors
    - Major difference
      - Focus the programmer takes during the earliest planning stages of a project

Summary

- Computer programming
  - Requires specific syntax
  - Must develop correct logic
- Programmer’s job
  - Understanding the problem, planning the logic, coding the program, translating the program into machine language, testing the program, putting the program into production, and maintaining it
- Procedural and object-oriented programmers approach problems differently