

**Chapter 1:
An Overview of Computers and
Logic**

**Programming Logic and
Design, 4th Edition Introductory**

Objectives

- **After studying Chapter 1, you should be able to:**
- **Understand computer components and operations**
- **Describe the steps involved in the programming process**
- **Describe the data hierarchy**
- **Understand how to use flowchart symbols and pseudocode statements**
- **Use and name variables**

Objectives (continued)

- **Use a sentinel, or dummy value, to end a program**
- **Use a connector symbol**
- **Assign values to variables**
- **Recognize the proper format of assignment statements**
- **Describe data types**
- **Understand the evolution of programming techniques**

Understanding Computer Components and Operations

- **Hardware:** equipment, or devices, associated with a computer
- For a computer to be useful, it needs more than equipment; a computer needs to be given instructions
- The instructions that tell the computer what to do are called **software**, or programs, and are written by programmers
- Hardware devices that perform **input** include keyboards and mice

Understanding Computer Components and Operations (continued)

- Through input devices,
 - **data**, or facts, enter the computer system
- **Processing** data items may involve
 - organizing them,
 - checking them for accuracy, or
 - performing mathematical operations on them

Understanding Computer Components and Operations (continued)

- The hardware that performs these sorts of tasks is the **central processing unit**, or **CPU**
- After data items have been processed, the resulting information is sent to a printer, monitor, or some other **output** device so people can view, interpret, and use the results

Understanding Computer Components and Operations (continued)

- You write computer instructions in a computer **programming language**, such as Visual Basic, Pascal, COBOL, C++, Java, or Fortran
- Every language has rules governing its word usage and punctuation
- Programming rules are called the language's **syntax**
- Each programming language uses a piece of software to translate the specific programming language into the computer's on-off circuitry, or **machine language**

Understanding Computer Components and Operations (continued)

- The language translation software, known as a **compiler** or **interpreter**, tells you if you have used a programming language incorrectly
- For a program to work properly, you must give the computer exact instructions in a specific sequence
- By doing this, you are developing the **logic** of the computer program
- Once instructions have been inputted to the computer and translated into machine language, a program can be **run**, or **executed**

Understanding Computer Components and Operations (continued)

- Besides input, processing, and output, all computer systems need and have:
 - **Internal storage**, commonly called memory, main memory, or primary memory. Though needed to run programs, internal memory is **volatile**—that is, its contents are lost every time the computer loses power
 - **External storage**, or permanent storage outside the main memory of the machine, is held on a device such as a floppy disk, hard disk, or magnetic tape

Understanding the Programming Process

- **The programmer's job can be broken down into six programming steps:**
 - 1. Understand the problem**
 - 2. Plan the logic**
 - 3. Code the program**
 - 4. Translate the program into machine language**
 - 5. Test the program**
 - 6. Debug**
 - 7. Put the program into production**

Understand The Problem

- **Really understanding the problem may be one of the most difficult aspects of programming**
 - **The description of what the user needs may be vague**
 - **The user may not even really know what he or she wants**
 - **Users who think they know what they want frequently change their minds after seeing sample output**
- **A good programmer is often part counselor, part detective**

Plan the Logic

- **Programmer plans the steps to the program, deciding what steps to include and how to order them**
- **The two most common tools are **flowcharts** and **pseudocode****
- **Both tools involve writing the steps of the program in English**

Code the Problem

- **Some very experienced programmers can successfully combine the logic planning and the actual instruction writing, or coding of the program, in one step**
- **A good term paper needs planning before writing, and so do most programs**

Translate the Program into Machine Language

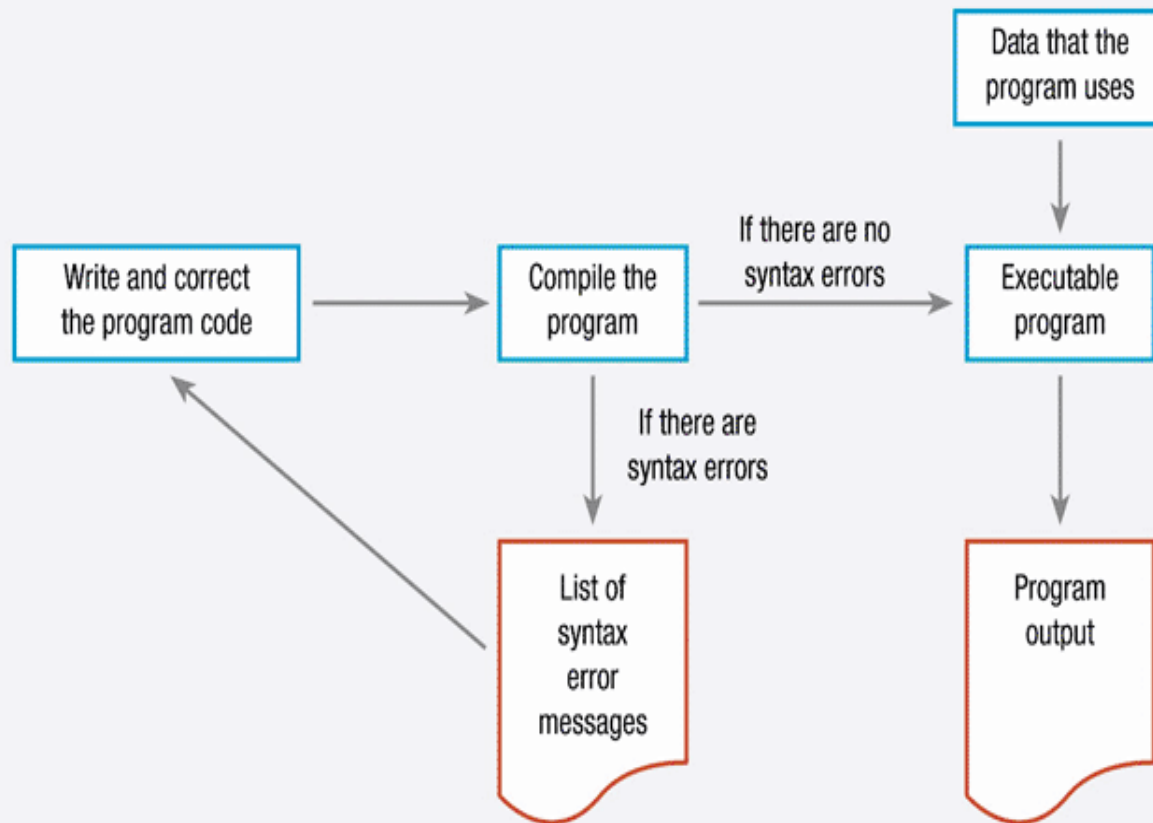
- Languages such as Java or Visual Basic translate the programmer's English-like **high-level programming language** into the **low-level machine language** that the computer understands
- If you write a programming language statement incorrectly (for example, by misspelling a word, using a word that doesn't exist in the language, or using "illegal" grammar), the translator program doesn't know what to do and issues an error message identifying a **syntax error**

Translate the Program into Machine Language (continued)

- All syntax errors are caught by the compiler or interpreter
- When writing a program, a programmer might need to recompile the code several times
- An executable program is created only when the code is free of syntax errors
- When you run an executable program, it might also typically require **input data**

Creating an Executable Program

FIGURE 1-1: CREATING AN EXECUTABLE PROGRAM



Test the Program

- A program that is free of syntax errors is not necessarily free of **logical errors**
- Once a program is free from syntax errors, the programmer can test it—that is, execute it with some sample data to see whether the results are logically correct

Put the Program into Production

- Putting a program into production might mean simply running the program once, if it was written to satisfy a user's request for a special list
- The process might take months if the program will be run on a regular basis, or it is one of a large system of programs being developed
- **Conversion**, the entire set of actions an organization must take to switch over to using a new program or set of programs, can sometimes take months or years to accomplish

Understanding the Data Hierarchy

- When data is stored for use on computer systems, it is often stored in a **data hierarchy**, where the smallest usable unit of data is the character
- **Characters** are letters, numbers, and special symbols, such as “A”, “7”, and “\$”
- A **field** is a single data item, such as `lastName`, `streetAddress`, or `annualSalary`

Understanding the Data Hierarchy (continued)

- **Records** are groups of fields that go together for some logical reason
- **Files** are groups of records that go together for some logical reason
- A **database** holds a group of files, often called **tables**, that together serve the information needs of an organization
- Database software establishes and maintains relationships between fields in these tables, so that users can write questions called **queries**

Using Flowchart Symbols and Pseudocode Statements

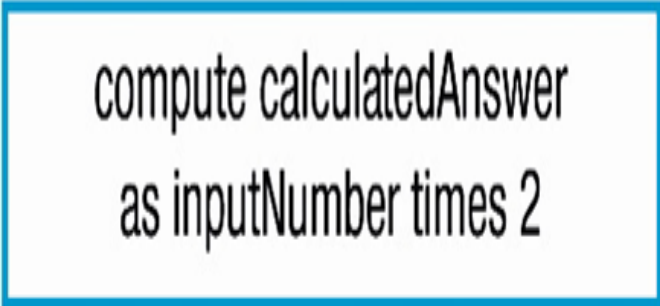
- **Flowcharts** (pictorial representations) and **pseudocode** (English-like representations) are used by programmers to plan the logical steps for solving a programming problem
- Some professional programmers prefer writing pseudocode to drawing flowcharts, because using pseudocode is more similar to writing final statements in programming language

Using Flowchart Symbols and Pseudocode Statements (continued)

- **Almost every program involves the steps of input, processing, and output, necessitating some graphical way to separate them**
- **Arithmetic operation statements are examples of processing in a flowchart, where you use a rectangle as the **processing symbol** containing a processing statement**

Using Flowchart Symbols and Pseudocode Statements (continued)

FIGURE 1-5: PROCESSING SYMBOL

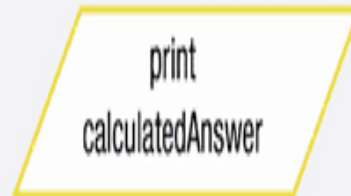


```
compute calculatedAnswer  
as inputNumber times 2
```

Using Flowchart Symbols and Pseudocode Statements (continued)

- To represent an **output** statement, you use the **parallelogram**, which is also the same symbol used for **input** statements

FIGURE 1-b: OUTPUT SYMBOL

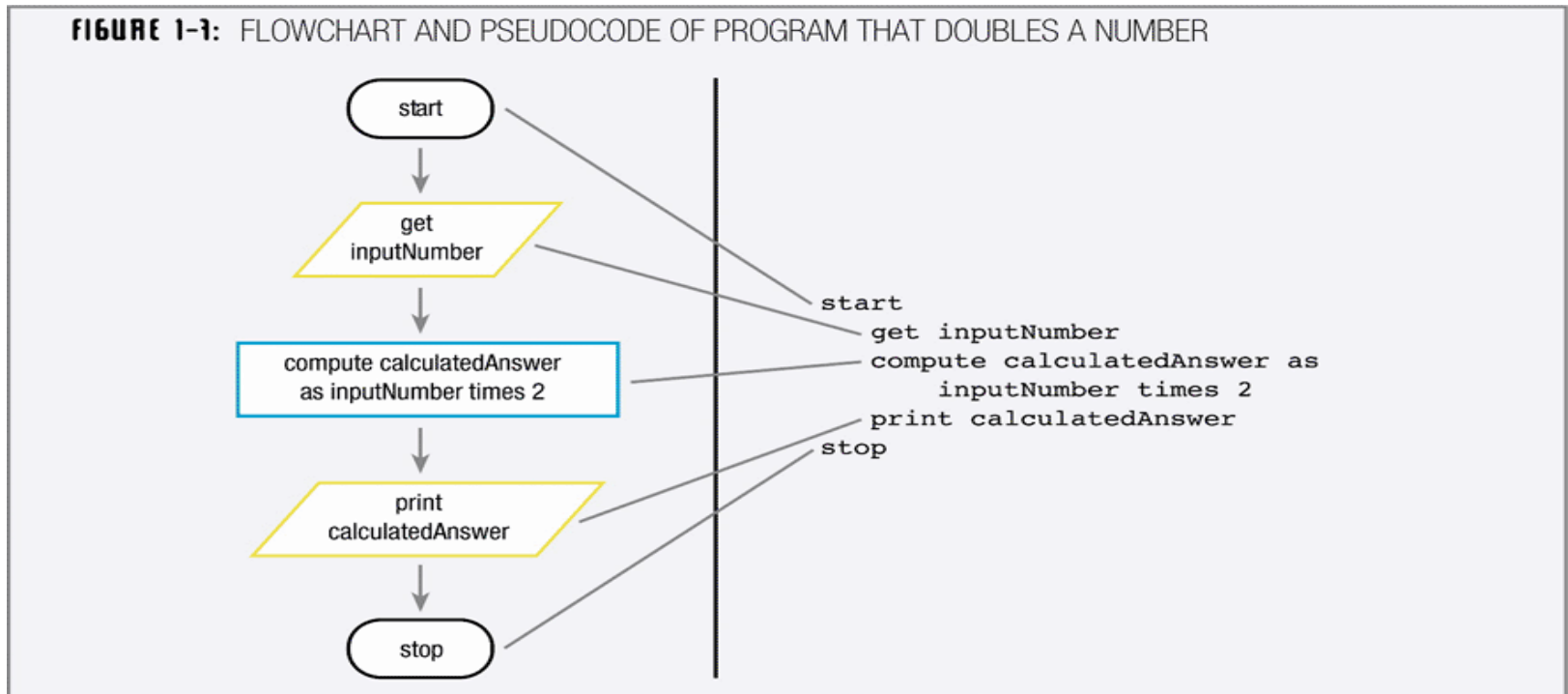


Using Flowchart Symbols and Pseudocode Statements (continued)

- In flowcharts:
 - Arrows, or **flowlines**, connect and show the appropriate sequence of steps
 - A **terminal symbol**, or start/stop symbol, should be included at each end
 - Often, “start” or “begin” is used as the first terminal symbol and “end” or “stop” is used in the other
 - The standard terminal symbol is shaped like a racetrack; often called a **lozenge**, because it resembles the shape of a medicated candy lozenge you might use to soothe a sore throat

Using Flowchart Symbols and Pseudocode Statements (continued)

- Figure 1-7 shows a complete flowchart for the program that doubles a number, and the pseudocode for the same problem



Using and Naming Variables

- **Variables** are memory locations, whose contents can vary or differ over time
- Sometimes, `inputNumber` can hold a 2 and `calculatedAnswer` will hold a 4; at other times, `inputNumber` can hold a 6 and `calculatedAnswer` will hold a 12
- A variable name is also called an **identifier**

Using and Naming Variables (continued)

- **Variable names used here follow only two rules:**
 - 1. Must be one word*
 - 2. Have some appropriate meaning*
- **Table 1-1 on page 19 of the text lists some possible variable names that might be used to hold an employee's last name and provides a rationale for the appropriateness of each one**

Ending a Program By Using Sentinel Values

- An **infinite loop** is a repeating flow of logic with no end
- To end the program, set a predetermined value for `inputNumber` that means “Stop the program!”
- The program can then test any incoming value for `inputNumber` and, if it is a 0, stop the program
- Testing a value is also called making a **decision**
 - Represented in flowchart by diamond shape called a **decision symbol**

Ending a Program By Using Sentinel Values (continued)

- A pre-selected value that stops the execution of a program is often called a **dummy value** since it does not represent real data, but just a signal to stop
- Sometimes, such a value is called a **sentinel value** because it represents an entry or exit point, like a sentinel who guards a fortress

Using the Connector

- By using just the input, processing, output, decision, and terminal symbols, you can represent the flowcharting logic for many diverse applications
- When drawing a flowchart segment, you might use only one other symbol, the **connector**
- You can use a connector when limited page size forces you to continue a flowchart in an unconnected location or on another page

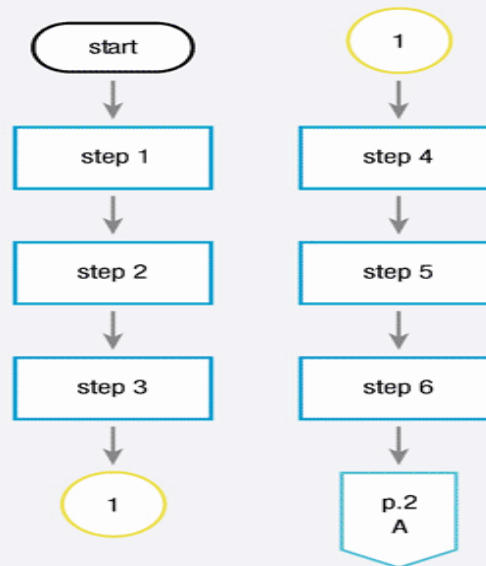
Using a Connector (continued)

- **By convention, programmers use a circle as an on-page connector symbol, and a symbol that looks like a square with a pointed bottom as an off-page connector symbol**

Using a Connector (continued)

- If a flowchart has six processing steps and a page provides room for only three, you might represent the logic as shown below:

FIGURE 1-12: FLOWCHART USING THE CONNECTOR



Assigning Values to Variables

- **When you create a flowchart or pseudocode for a program that doubles numbers, you can include the statement `compute calculatedAnswer as inputNumber times 2`**
- **This statement incorporates two actions:**
 - **First, the computer computes the arithmetic value of `inputNumber times 2`**
 - **Second, the computed value is stored in the `calculatedAnswer` memory location**

Assigning Values to Variables (continued)

- Most programming languages allow a shorthand expression for **assignment statements** such as `compute calculatedAnswer as inputNumber times 2`
- The shorthand takes the form `calculatedAnswer = inputNumber * 2`
- The equal sign is the **assignment operator**, which always requires the name of a memory location on its left side—the location where the result will be stored

Understanding Data Types

- Computers deal with two basic types of data—character and numeric
- When you use a specific numeric value, such as 43, within a program, you write it using the digits and no quotation marks
- A specific numeric value is often called a **numeric constant** because it does not change—a 43 always has the value 43
- When you use a specific character value, or **string** of characters, such as “Chris” you enclose the string, or **character constant**, within quotation marks

Understanding Data Types (continued)

- **Most computer languages allow at least two distinct types of variables:**
 - One holds a number, often called a **numeric variable**
 - Others hold letters of the alphabet and various special characters such as punctuation marks, and are called **character, text, or string variables**, depending on the language being used

Understanding Data Types (continued)

- **Some languages allow for several types of numeric data**
- **Languages such as Pascal, C++, C#, and Java distinguish between **integer** (whole number) numeric variables and **floating-point** (fractional) numeric variables containing a decimal point**

Understanding the Evolution of Programming Techniques

- **Old programming languages required programmers to work with memory addresses and to memorize awkward codes associated with machine languages**
- **Newer programming languages look much more like natural language and are easier to use**

Understanding the Evolution of Programming Techniques (continued)

- **Currently, there are two major techniques used to develop programs and their procedures**
 - **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

Summary

- **A programmer's job involves:**
 - **Understanding the problem**
 - **Planning the logic**
 - **Coding the problem**
 - **Translating the program into machine language**
 - **Testing the program**
 - **Putting the program into production**
- **When programmers plan the logic for a solution to a programming problem, they often use flowcharts or pseudocode**

Summary (continued)

- **Testing a value involves making a decision**
- **Most programming languages use the equal sign to assign values to variables**
- **Procedural and object-oriented programmers approach program problems differently**