



Programming Logic and Design

Seventh Edition

Chapter 1

An Overview of Computers and

Programming



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
 - Tells the hardware what to do
 - **Programs**
 - Instructions written by programmers

Understanding Computer Systems (continued)

- **Application software** such as word processing, spreadsheets, payroll and inventory, even games
- **System software** such as operating systems like Windows, Linux, or UNIX
- Computer hardware and software accomplish three major operations
 - **Input**
 - **Data items** such as text, numbers, images, and sound
 - **Processing**
 - Calculations and comparisons performed by the **central processing unit (CPU)**

Understanding Computer Systems (continued)

- **Output**
 - Resulting information that is sent to a printer, a monitor, or **storage devices** after processing
- **Programming language**
 - Used to write computer instructions
 - Examples
 - Visual Basic, C#, C++, or Java
- **Syntax**
 - Rules governing word usage and punctuation

Understanding Computer Systems (continued)

- **Computer memory**
 - Computer's temporary, internal storage – **random access memory (RAM)**
 - **Volatile** memory – lost when the power is off
- Permanent storage devices
 - **Nonvolatile** memory
- **Compiler or interpreter**
 - Translates source code into **machine language (binary language)** statements called **object code**
 - Checks for syntax errors

Understanding Simple Program Logic

- Program **executes** or **runs**
 - Input will be accepted, some processing will occur, and results will be output
- Programs with syntax errors cannot execute
- **Logical errors**
 - Errors in program logic produce incorrect output
- **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)

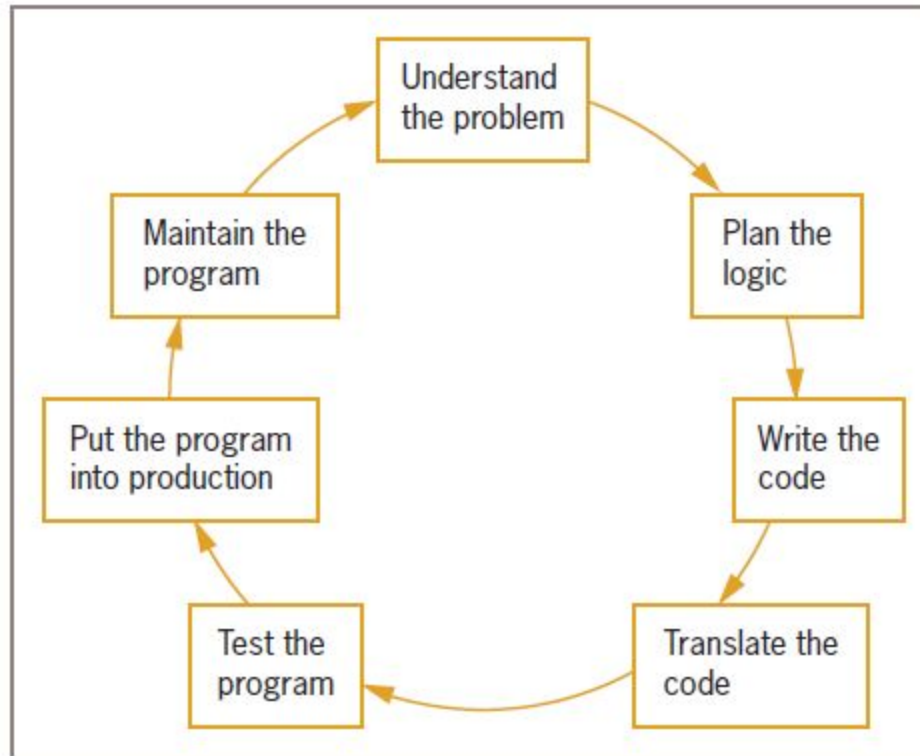


Figure 1-1 The program development cycle



Understanding the Problem

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



Planning the Logic

- Heart of the programming process
- Most common planning tools
 - Flowcharts
 - Pseudocode
 - **IPO charts** (input, processing, and output)
 - **TOE charts** (tasks, objects, and events)
- **Desk-checking**
 - Walking through a program's logic on paper before you actually write the program



Coding the Program

- Hundreds of programming languages available
 - Choose based on features
 - Similar in their basic capabilities
- Easier than the 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

Using Software to Translate the Program into Machine Language (continued)

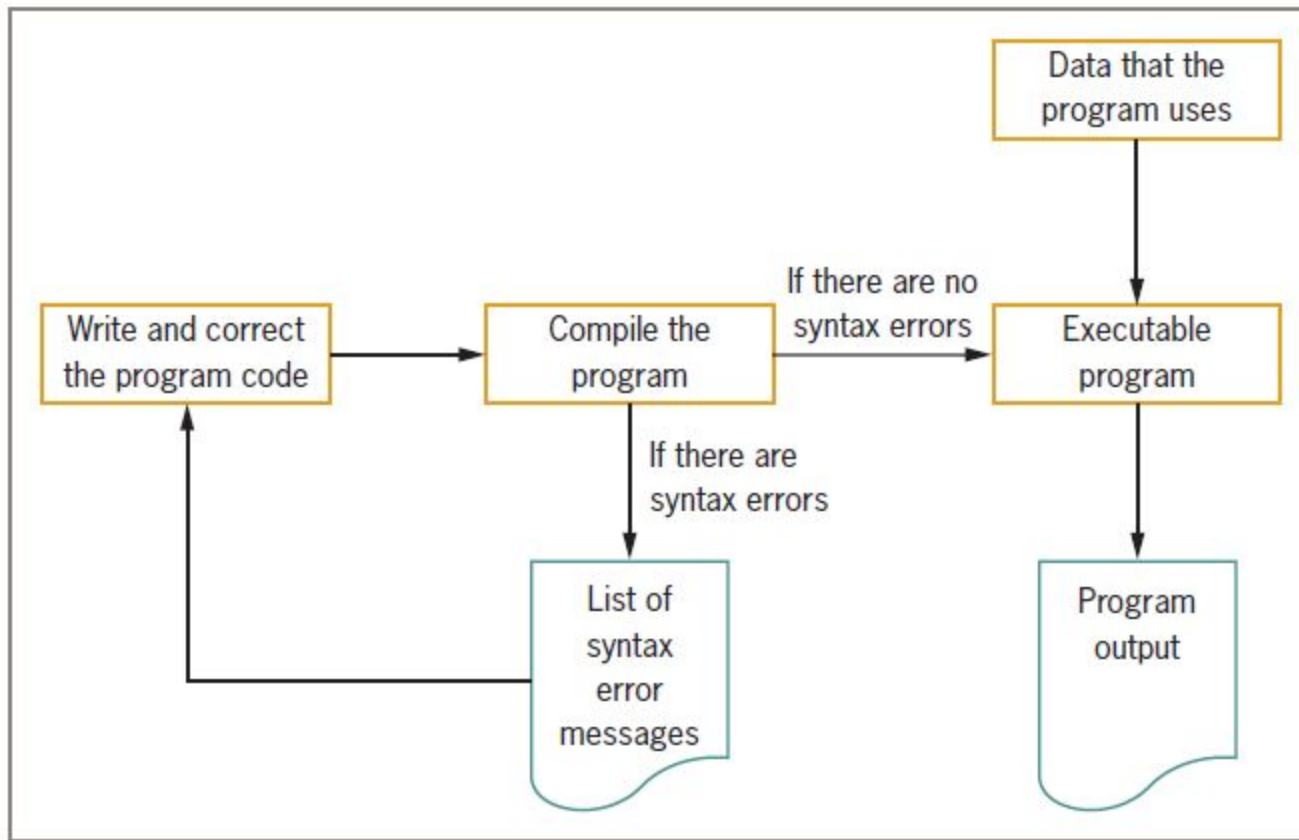


Figure 1-2 Creating an executable program



Testing the Program

- Logical error
 - Results when a syntactically correct statement, but the wrong one for the current context, is used
- Test
 - Execute the program with some sample data to see whether the results are logically correct
- **Debugging** is the process of finding and correcting program errors
- 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**
 - The 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 the 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
```

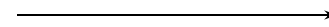


Writing Pseudocode (continued)

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

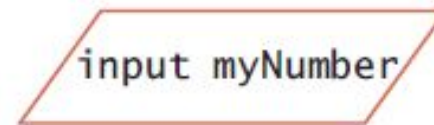
Drawing Flowcharts

- Create a flowchart
 - Draw geometric shapes that contain the individual statements
 - Connect shapes with arrows



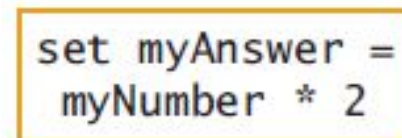
- **Input symbol**

- Indicates input operation
- Parallelogram



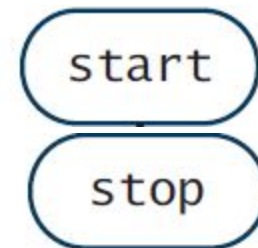
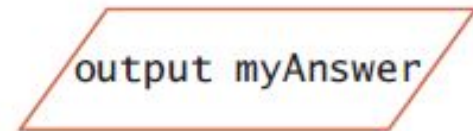
- **Processing symbol**

- Contains 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 lozenges



Drawing Flowcharts (continued)

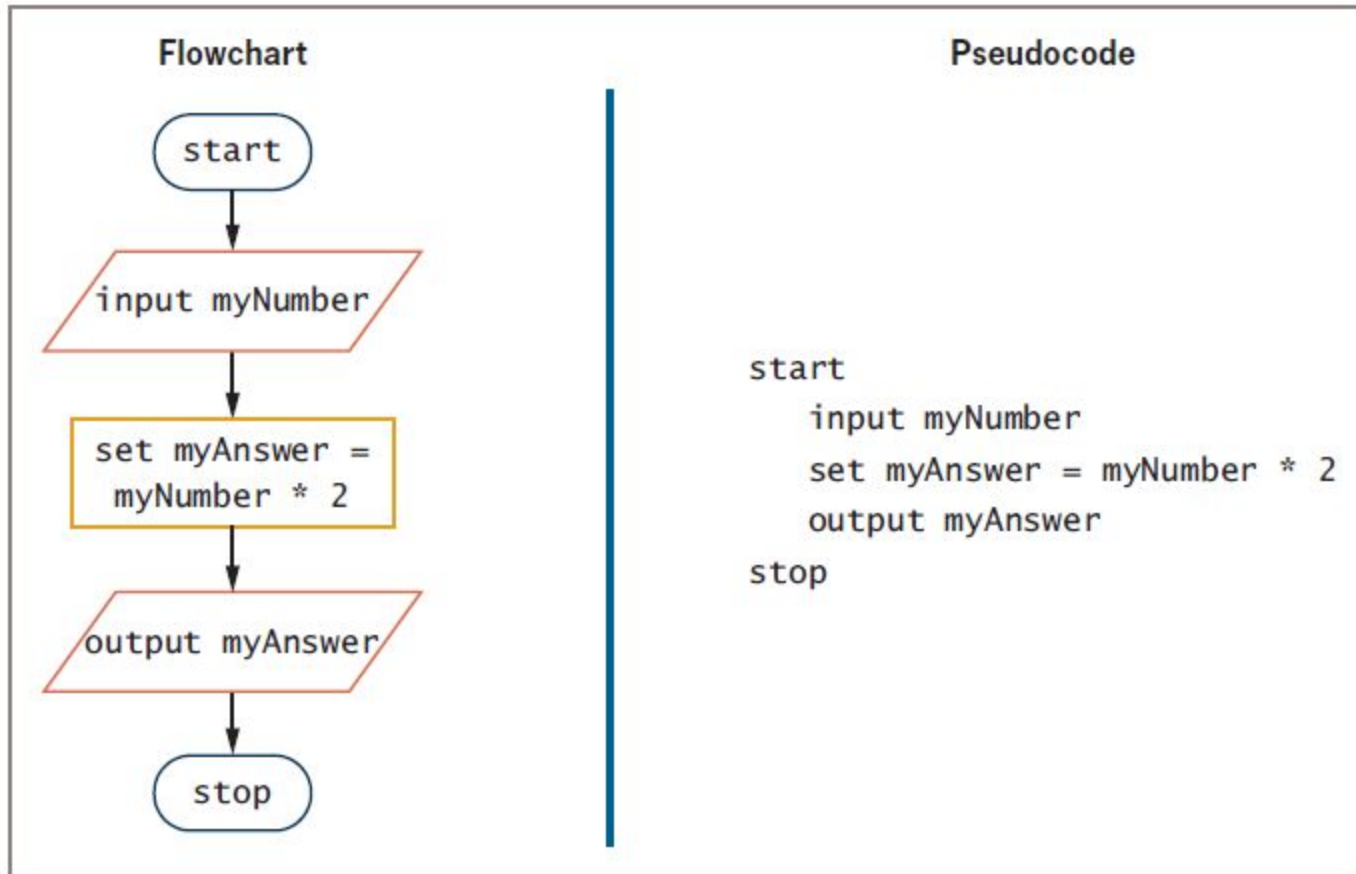


Figure 1-6 Flowchart and pseudocode of program that doubles a number



Repeating Instructions

- Program in Figure 1-6 only works for one number
- Not feasible to run the program over and over 10,000 times
- Not feasible to add 10,000 lines of code to a program
- Create a **loop** (repetition of a series of steps) instead
- Avoid an **infinite loop** (repeating flow of logic that never ends)

Repeating Instructions (continued)

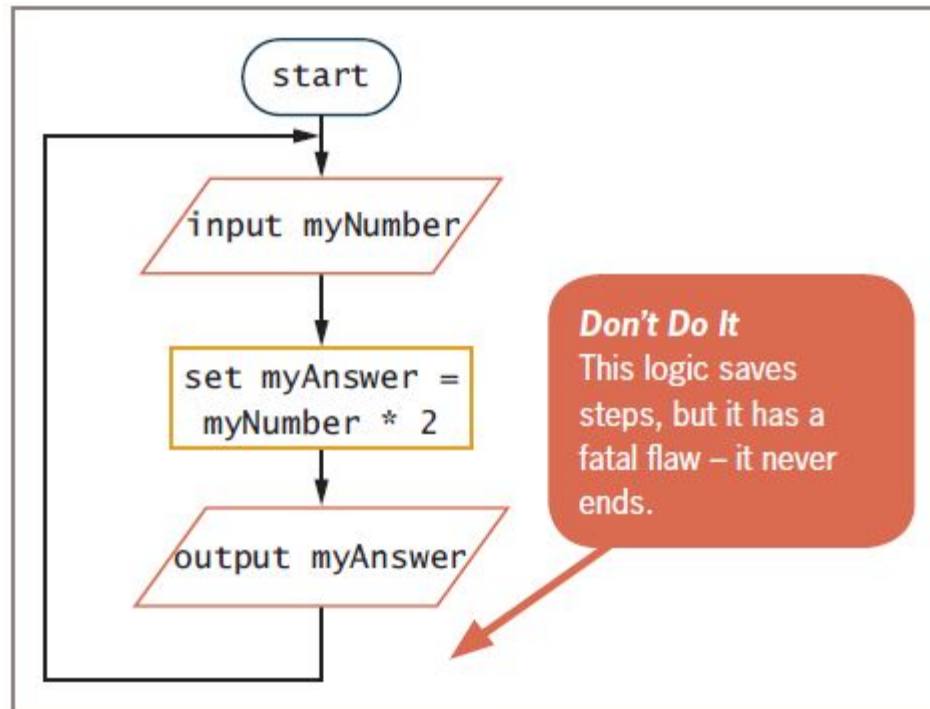
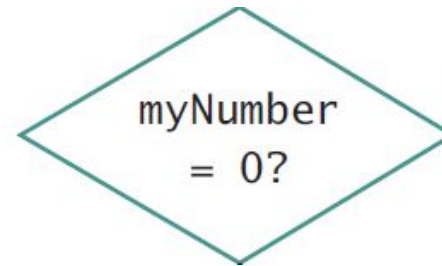


Figure 1-8 Flowchart of infinite number-doubling program

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

Using a Sentinel Value to End a Program (continued)

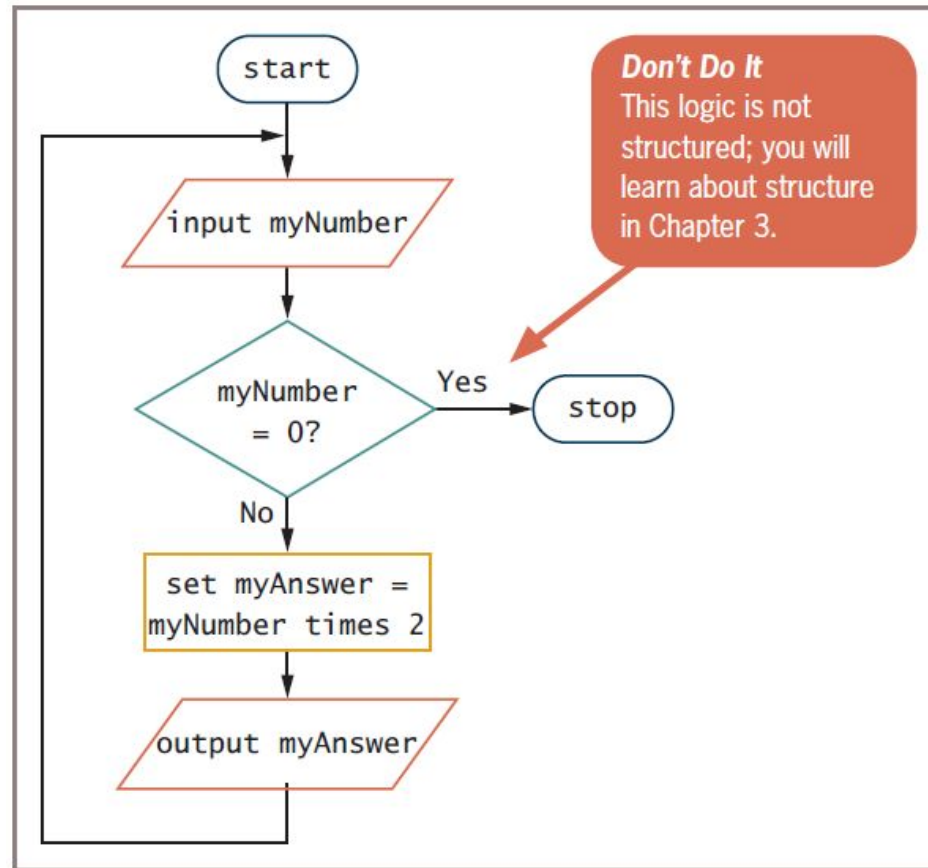


Figure 1-9 Flowchart of number-doubling program with sentinel value of 0

Using a Sentinel Value to End a Program (continued)

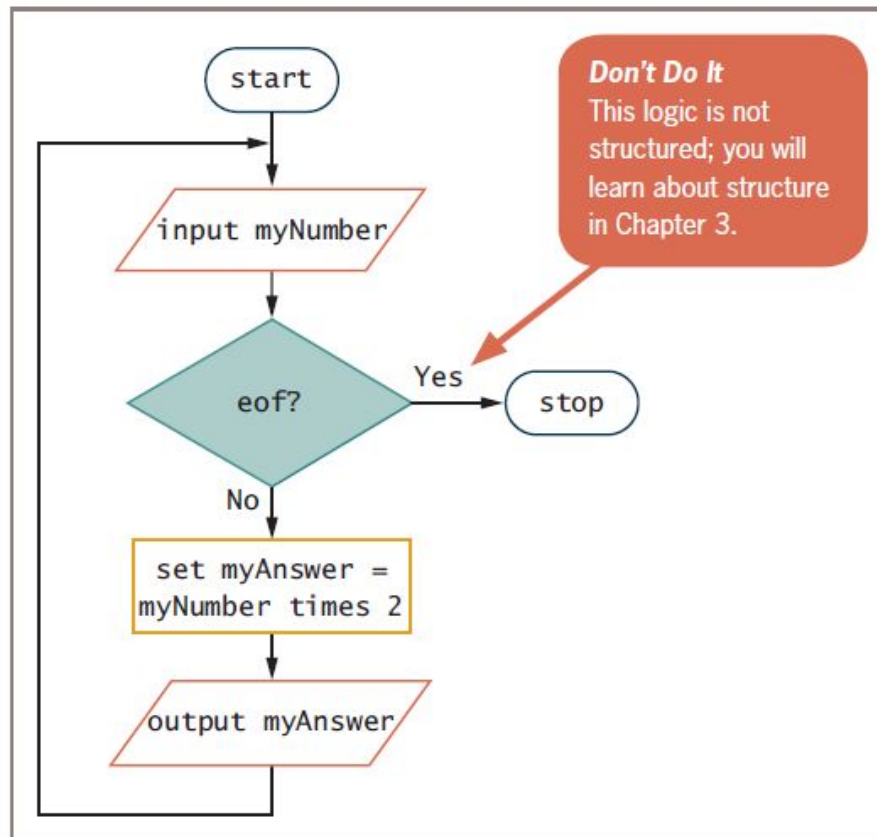


Figure 1-10 Flowchart using eof

Understanding Programming and User Environments

- Many options for programming and user environments
 - Planning
 - Flowchart
 - Pseudocode
 - Coding
 - Text editors
 - Executing
 - Input from keyboard, mouse, microphone
 - Outputting
 - Text, images, sound

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, a compiler, and other programming tools

Understanding Programming Environments (continued)

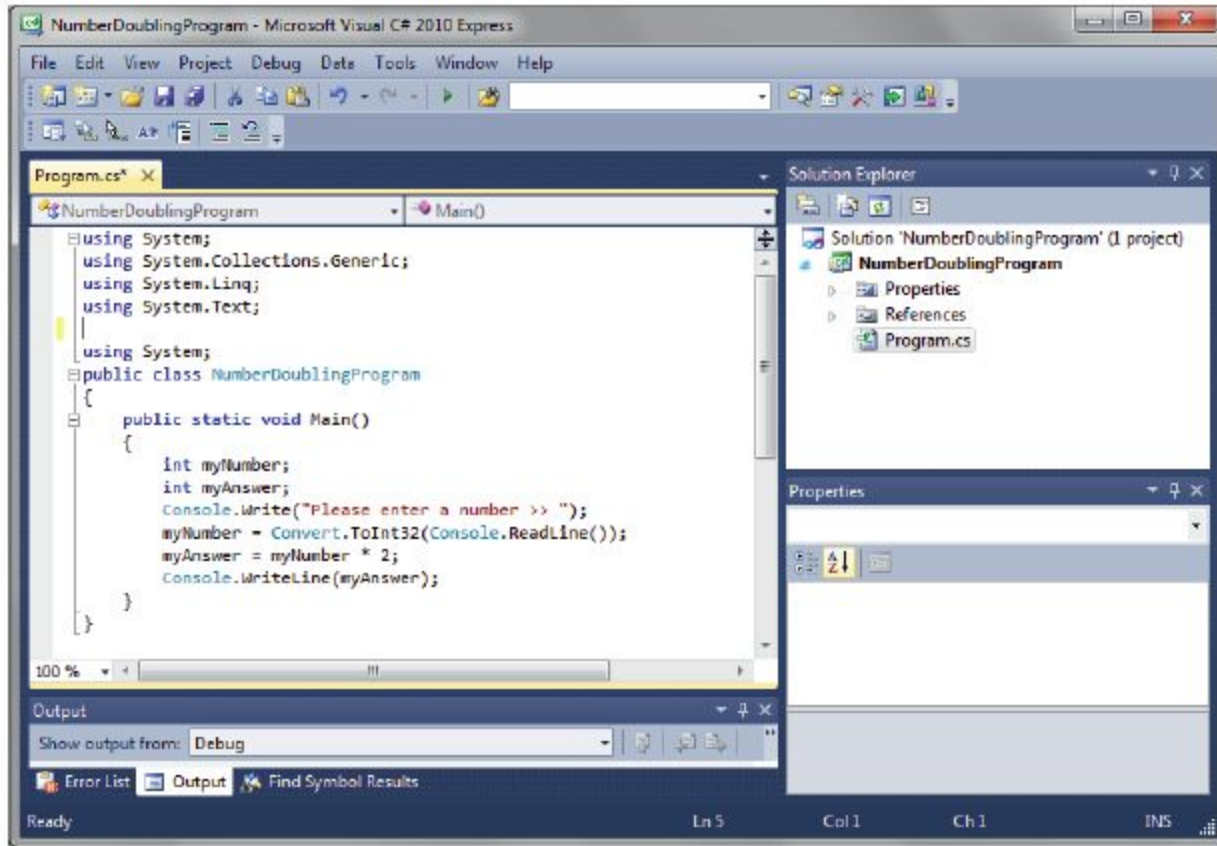
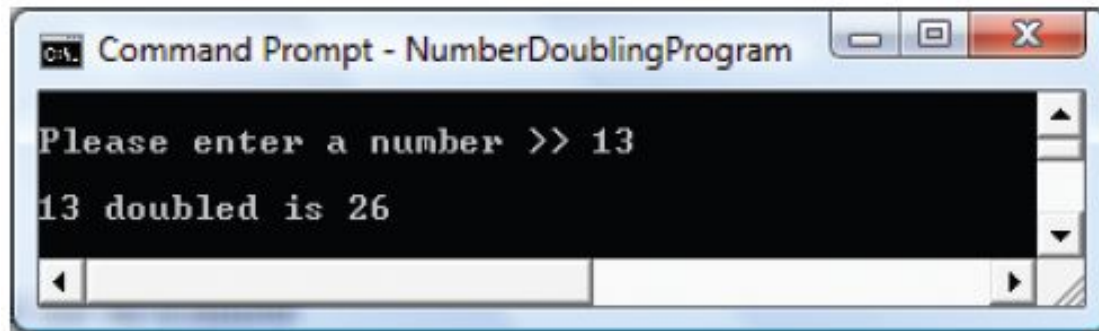


Figure 1-12 A C# number-doubling program in Visual Studio

Understanding User Environments

- **Command line**
 - Location on your computer screen where 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 User Environments (continued)



```
cmd: Command Prompt - NumberDoublingProgram
Please enter a number >> 13
13 doubled is 26
```

Figure 1-13 Executing a number-doubling program
in a command-line environment

Understanding User Environments (continued)

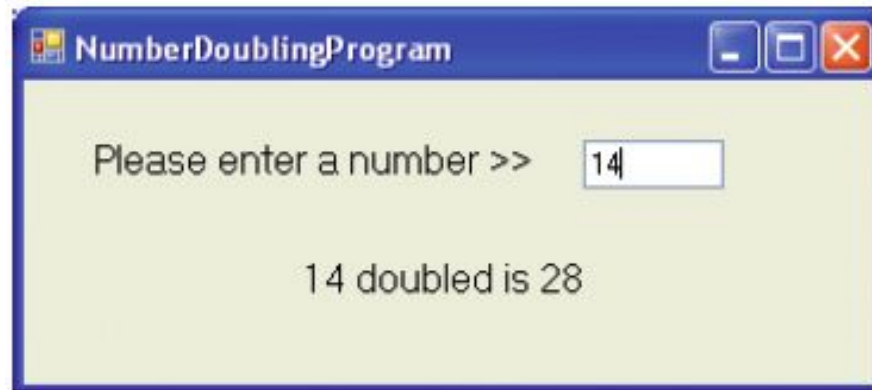


Figure 1-14 Executing a number-doubling program
in a GUI environment

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
 - Are 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
 - This text
 - Focuses on procedural programming techniques



Summary

- Hardware and software accomplish input, processing, and output
- Logic must be developed correctly
- Logical errors are much more difficult to locate than syntax errors
- Use flowcharts, pseudocode, IPO charts, and TOE charts to plan the logic
- Avoid infinite loops by testing for a sentinel value
- Use a text editor or an IDE to enter your program statements