Programming Logic and Design
Ninth Edition

Chapter 1

An Overview of Computers and Programming

Objectives

In this chapter, you will learn about:

- Computer systems
- Basic concepts about programming
- The steps involved in the program development cycle
- Pseudocode statements and flowchart symbols
- 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 that tell the hardware what to do

Understanding Computer Systems (continued -1)

- 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**)
  - **Output**
    - Resulting **information** that is sent to a printer, a monitor, or **storage devices** after processing
    - A **cloud** based device is accessed through the Internet

Computer Software

- Computer Software
  - **System software** such as operating systems like Windows, Linux, or UNIX, Google Android and Apple iOS
  - **Application software** such as word processing, spreadsheets, payroll and inventory, games, apps, etc.

Programming: Concepts

- **Programs**
  - Instructions written by programmers
- **Programming**
  - Writing software instructions
- **Programming language**
  - Used to write computer instructions called **program code**
  - Writing instructions is called **coding the program**
  - Examples
    - Visual Basic, C#, C++, or Java

Programming: Concepts (continued -1)

- **Syntax**
  - Rules governing word usage and punctuation
  - Mistakes in a language's usage are **syntax errors**
  - Programs with syntax errors cannot execute
- **Logic** of the computer program
  - Sequence of specific instructions in specific order
- **Logical errors**
  - Errors in program logic produce incorrect output

Programming: Concepts (continued -2)

- **Variable**
  - Named memory location whose value can vary
- **Computer memory**
  - Computer's temporary, internal storage – **random access memory** (**RAM**)
  - **Volatile** memory – lost when the power is off
- Permanent storage devices
  - **Nonvolatile** memory
  - Examples: hard drive, CD/DVD, jump drive, etc.

Programming: Concepts (continued -3)

- **Compiler** or **interpreter**
  - Translates **source code** into **machine language** (**binary language**) statements called **object code**
  - Checks for syntax errors
- Program **executes** or **runs**
  - Input will be accepted, some processing will occur, and results will be output

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 -1)

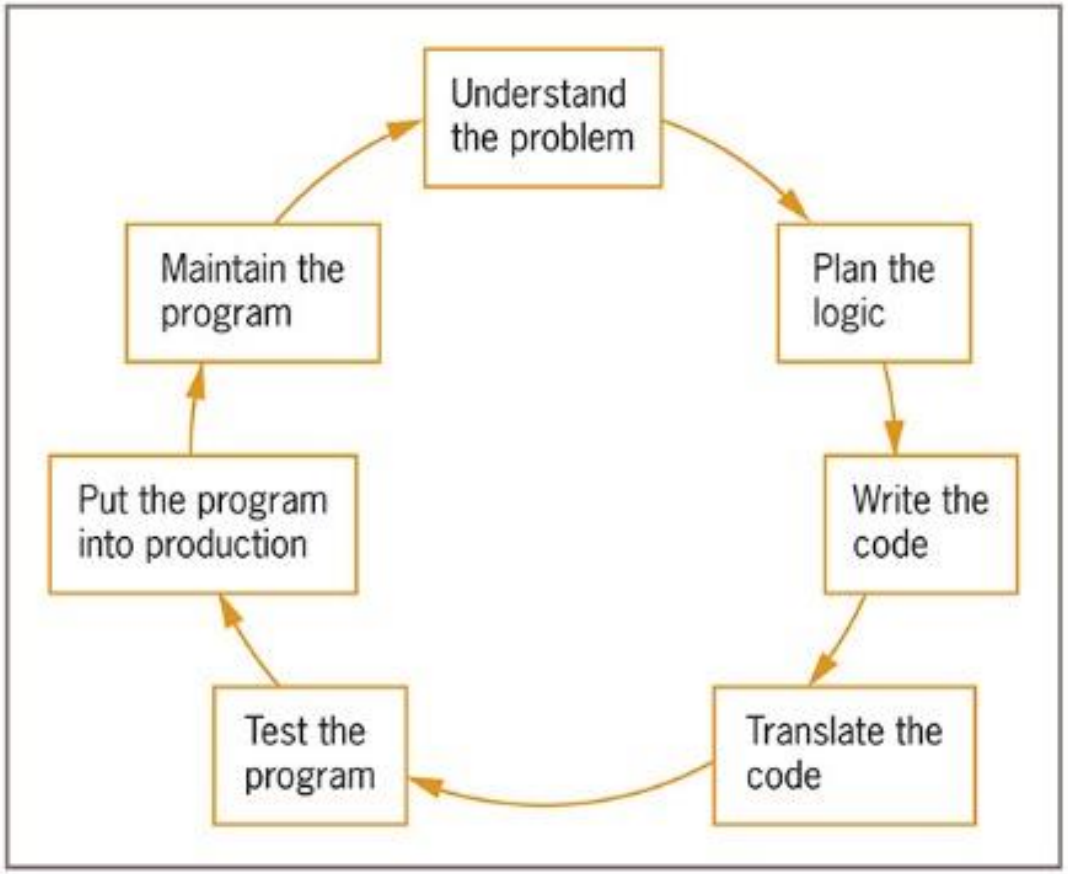


Figure 1-2 The program development cycle

Understanding the Problem

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

Planning the Logic

- Plan the steps of the program and what they include
- An **algorithm** is the sequence of steps or rules you follow to solve a problem
- 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 available
  - Choose based on features
  - Similar in their basic capabilities
- Coding is easier than the planning step
- Experienced programmers can successfully combine logic planning and program coding in one 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 -1)

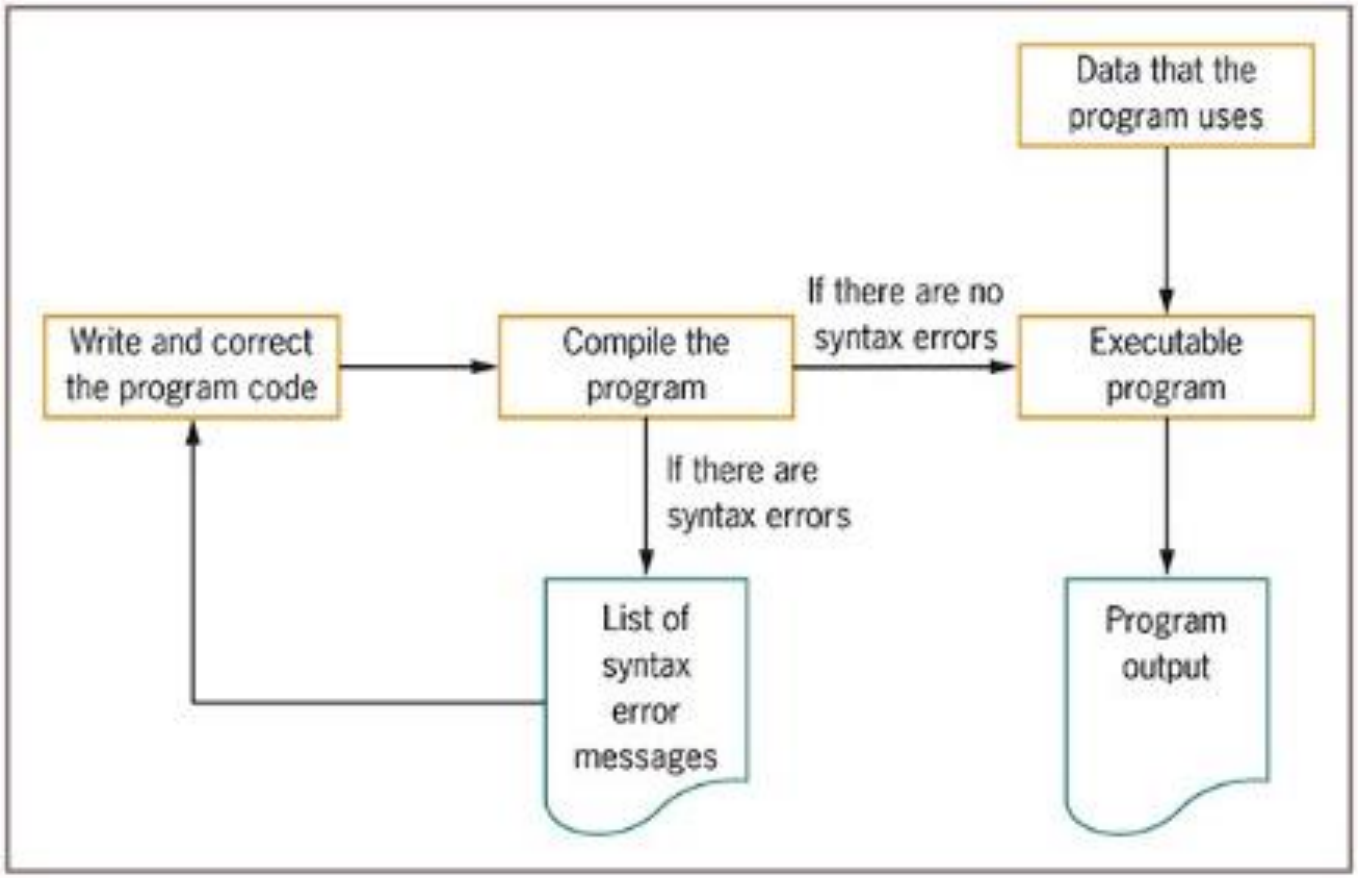


Figure 1-3 Creating an executable program

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
    - Training data-entry people on how to input data
    - Training users on how to use and understand output
    - Modify existing data to fit for the new format
- **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
```

Pseudocode Standards

- Programs begin with the word ***start*** and end with the word ***stop***; these two words are always aligned
- Whenever a module name is used, it is followed by a set of parentheses
- Modules begin with the module name and end with return. The module name and return are always aligned
- Each program statement performs one action—for example, input, processing, or output

Pseudocode Standards (continued -1)

- Program statements are indented a few spaces more than the word start or the module name
- Each program statement appears on a single line if possible. When this is not possible, continuation lines are indented
- Program statements begin with lowercase letters
- No punctuation is used to end statements

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

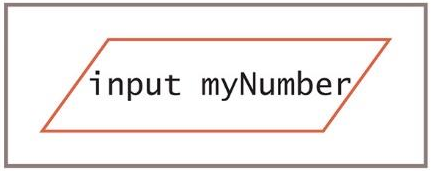


Figure 1-4 Input symbol

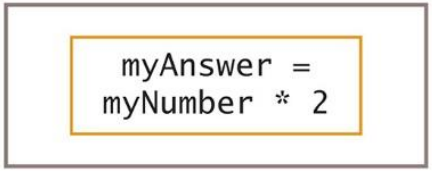
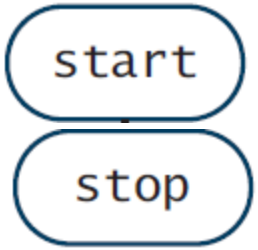


Figure 1-5 Processing symbol

Drawing Flowcharts (continued -1)

- **Output symbol**
  - Represents output statements
  - Parallelogram
- **Flowlines**
  - Arrows that connect steps
- **Terminal symbols**
  - Start/stop symbols
  - Shaped like a racetrack
  - Also called lozenges

Figure 1-6 Output symbol



Drawing Flowcharts (continued -2)

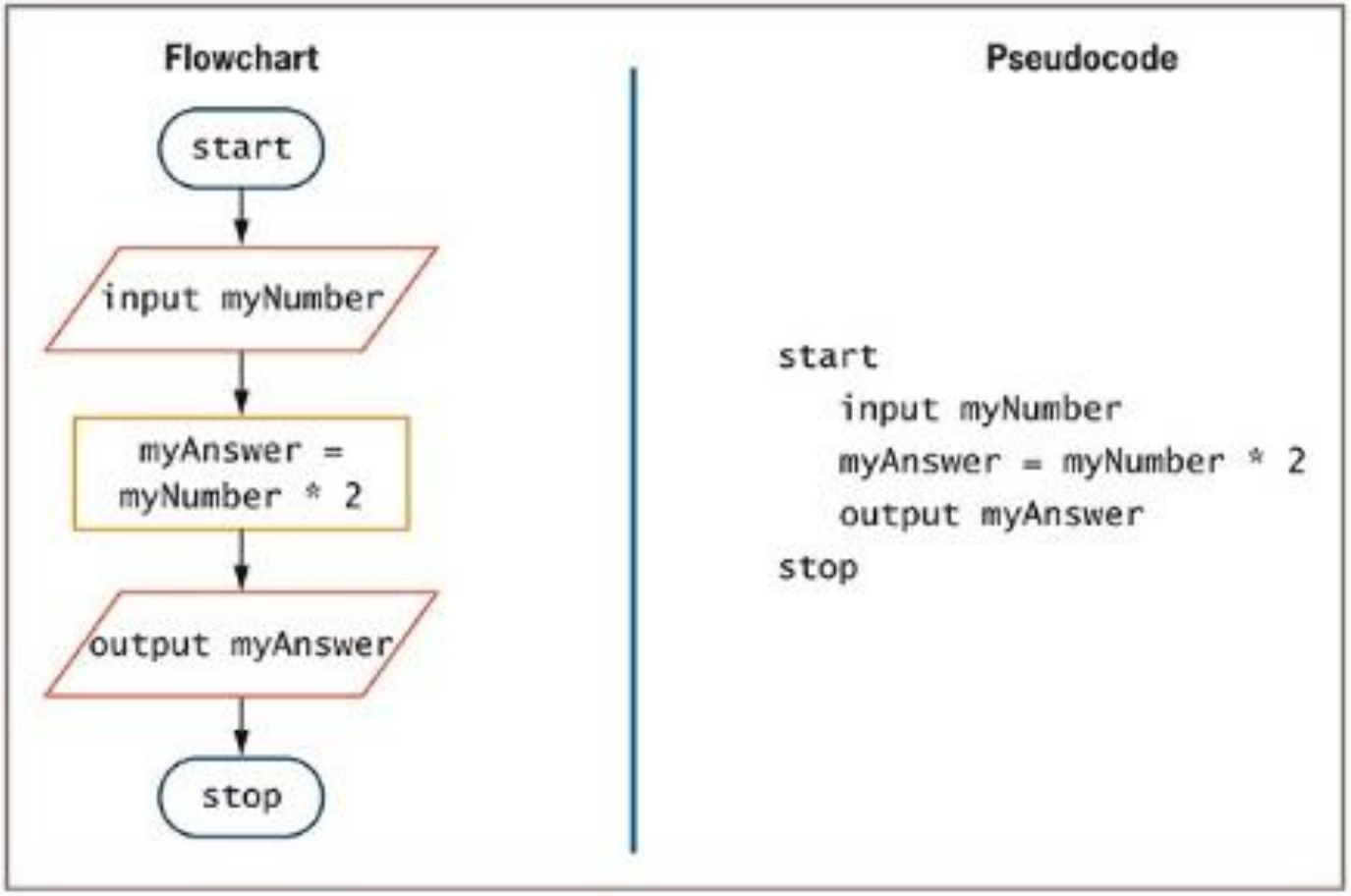


Figure 1-7 Flowchart and pseudocode of program that doubles a number

Understanding Programming Environments

- **Understanding Programming Environments**
  - **Text Editor** is used to create simple text files
  - **Integrated development environment (IDE)** provides an editor, compiler, and other programming tools
    - Microsoft Visual Studio IDE

Understanding Programming Environments (continued -1)

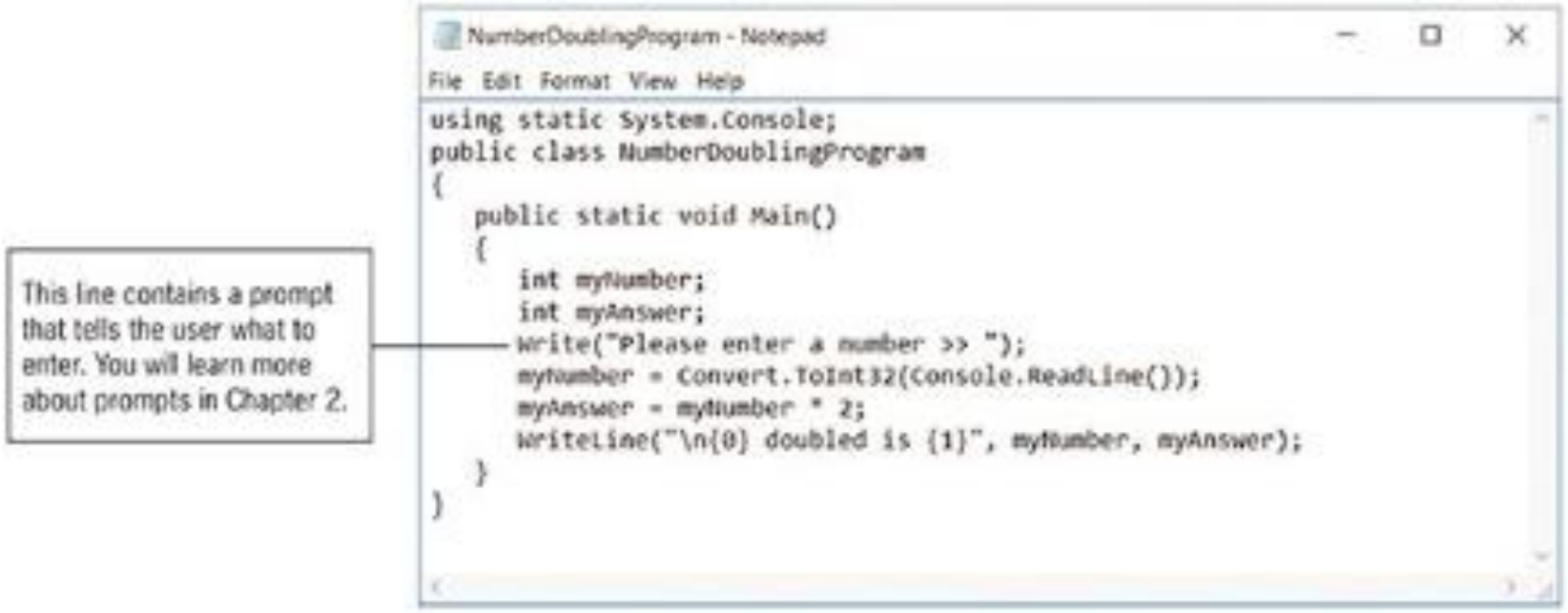


Figure 1-12 A C# number-doubling program in Notepad

Understanding Programming Environments (continued -2)

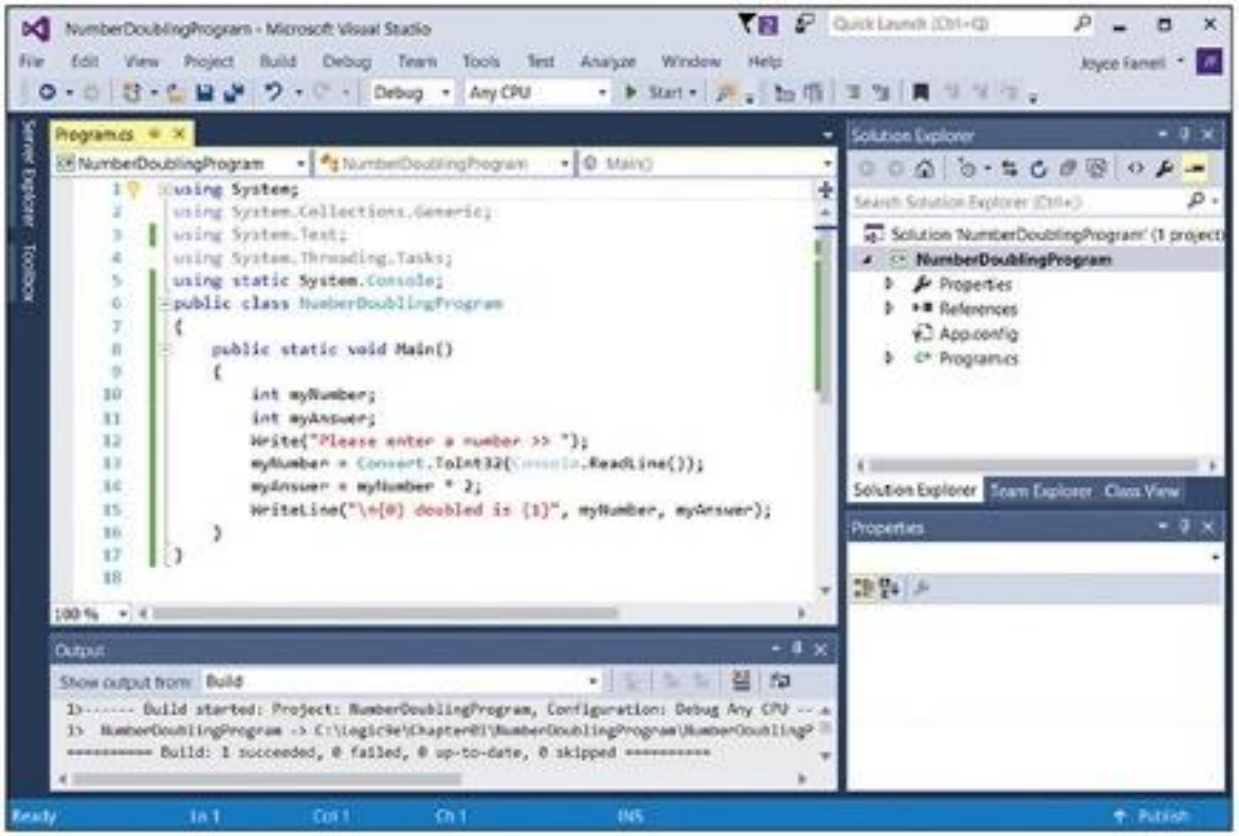


Figure 1-13 A C# number-doubling program in Visual Studio

Understanding User Environments

- **Understanding User Environments**
  - **Command line** is a location on your computer screen at which you type text entries to communicate with the computer's operating system
  - A **graphical user interface**, or **GUI** (pronounced gooey), allows users to interact with a program in a graphical environment

Understanding User Environments (continued -1)

Figure 1-14 Executing a number-doubling program in a command-line environment

Understanding User Environments (continued -2)

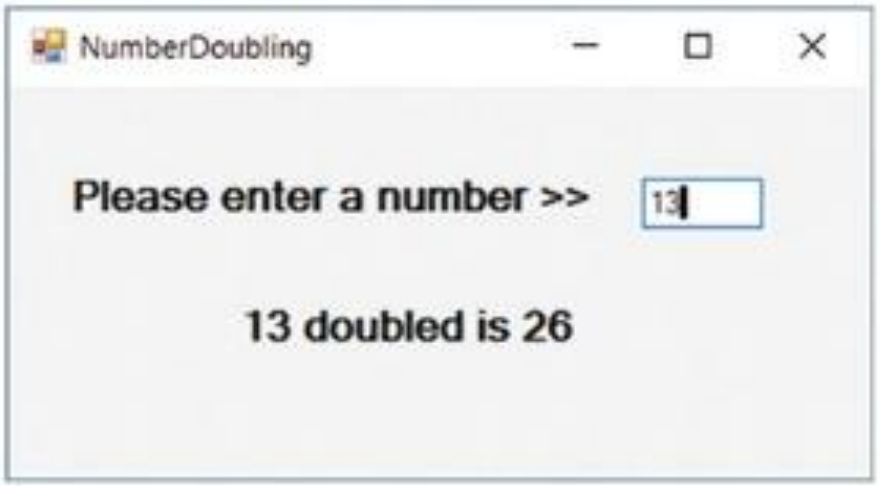


Figure 1-15 Executing a number-doubling program in a GUI environment

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 -1)

- 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

Summary

- Computer system
  - Hardware and software work together to accomplish input, processing, and output
  - Computer software: system and application
- Programming
  - The program development cycle
  - Pseudocode and flowchart for planning the logic
  - Programming and user environments
  - The evolution of programming models

Thank You!