1 Introduction to Computing and Application Development

C# Programming:
From Problem Analysis to Program Design, 5th Edition

Chapter Objectives

1. Investigate the steps of software development
2. Explore different programming methodologies
3. Discover why C# is being used today for software development
4. Distinguish between the different types of applications that can be created with C#
5. Explore an application written in C#
6. Examine the basic elements of a C# program
7. Compile, run, build, and debug an application
8. Create an application that displays output
9. Work through a programming example that illustrates the chapter's concepts

Software and Hardware

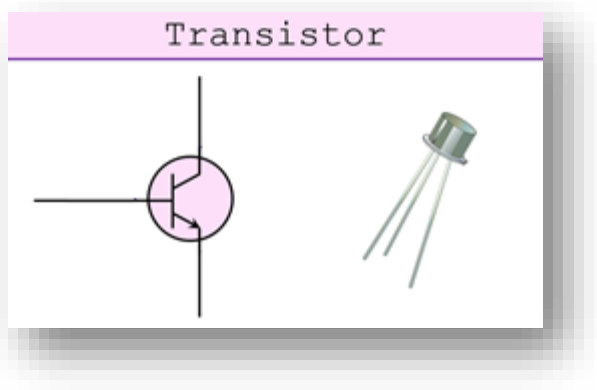
Any computing system (computer, cell phone, computing in airplane, cars, watching machine…) has 2 parts:

1. Software consists of programs
   - Sets of instructions telling the computer exactly what to do
2. Hardware consists of electronics

- Analogy: physical brain is **Hardware**, our thoughts/ideas are **Software**
- In this course, our focus is on Software

Machine Language

- Computer (machine) understands ONLY 0 and 1
- All software will be in 0 and 1 at the end
  - Because computer understands only 0 and 1
- The digits 1 and 0 used in binary reflect the on and off states of a transistor.



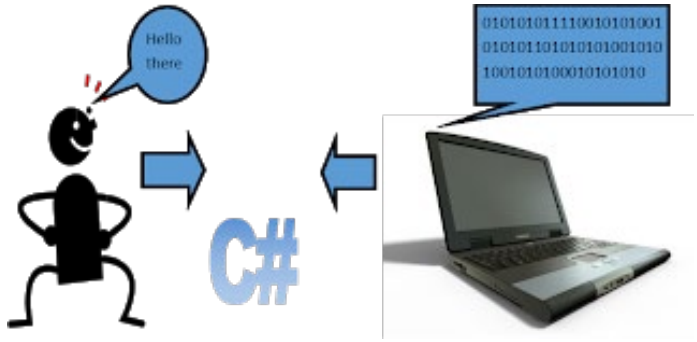
Machine language instructions

10110010	10011001	
10001100	11000100	11000110
10111100	01001100	11011110
10001100	01000101	
11110100	00101100	11111110
10101000	11000100	11000001

Human Language Vs. Machine Language

- Programmer is not a machine!
- Programmer knows Human Language (English)

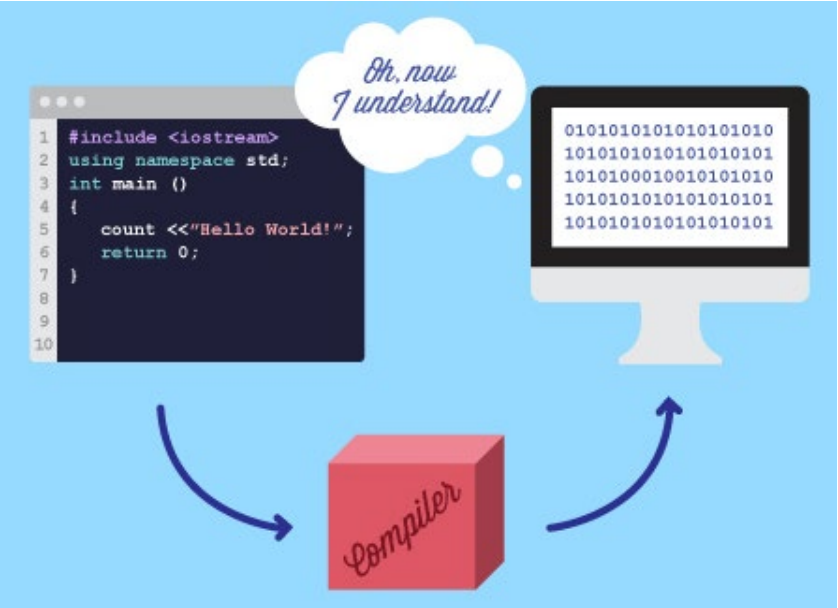
- Computer is not a human being!
- Computer understands Machine Language!



- How can they talk to each other?
  - Yes, we need
    1. A structured language like C#
    2. And a translator that is called a Compiler

Compiler

- Compiler is a software (a translator software)
  - First, checks syntax (grammar) errors of programmer
  - Then, translates High Level Languages (e.g. C#) to machine code
  - Visual Studio Includes a compiler



Software Development Process

- Programming is a process of problem solving
- How do you start?
    1. Analyze the problem
    2. Design a solution
    3. Code the solution
    4. Implement the code
    5. Test and debug

Steps in the Program Development Process (con

- Software development process is iterative
- As errors are discovered, it is often necessary to cycle back to previous phase or step

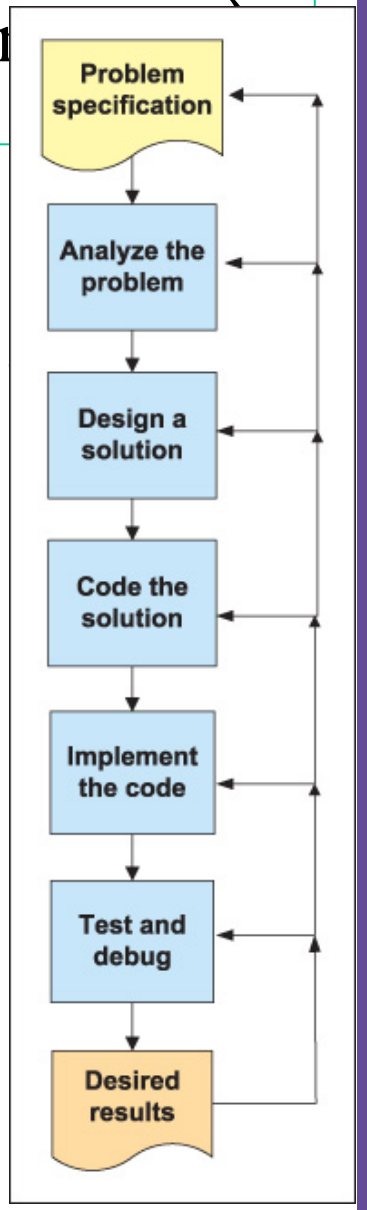


Figure 1-2 Steps in the software development process

Step 1: Analyze the Problem

- Precisely, what is software supposed to accomplish?
- Understand the problem definition
- Review the problem specifications

Analyze the Problem (continued)

Rapid Ready Car Rental Agency rents four types of vehicles:

Economy
Intermediate
Full size
Specialty-sports

The economy rents for $31.95 per day; the intermediate rents for $41.95 per day; the full size rents for $49.95 per day; and the specialty-sports rents for $59.95 per day.
They offer a 10% discount for rental periods in excess of 7 days. Rapid Ready has a policy that prohibits rental for periods beyond 30 days.
Allow the user to select the type of vehicle and number of total days before displaying the final price.

FIGURE 1-3 Program specification sheet for a car rental agency problem

Analyze the Problem (continued)

- What kind of data will be available for input?
- What types of input values (i.e., whole numbers, alphabetic characters, and numbers with decimal points) will be in each of the identified data items?
- What is the domain of input values (range of the values)?
- Will the user of the program be inputting values?
- If the problem solution is to be used with multiple data sets, are there any data items that stay the same, or remain constant, with each set?

Analyze the Problem (continued)

- Example: Analyze of the car rental problem (sample input for each data item)

Data identifier	Data type	Domain of values
kindOfVehicle	char (single coded character)	E, I, F, or S
noOfDays	Integer (whole number)	1...30

FIGURE 1-4 Data for car rental agency

Step 2: Design a Solution

- Several methodologies
  - Procedural
  - object-oriented (we follow this method)
- Divide and Conquer
  - Break the problem into smaller subtasks
  - Top-down design, stepwise refinement

Design a Solution (continued)

- Algorithm
  - Clear, unambiguous, step-by-step process for solving a problem
  - Steps must be expressed so completely and so precisely that all details are included
  - Instructions should be simple to perform
  - Instructions should be carried out in a finite amount of time
  - Following the steps blindly should result in the same results

Step 3: Code the Solution

- After completing the design, verify the algorithm is correct
- Translate the algorithm into *source code*
  - Follow the rules of the language (syntax)
- Integrated Development Environment (IDE): Tools for typing program statements, compiling, executing, and debugging applications
- Examples:
  - Visual Studio, Eclipse, Android Studio, NetBeans, JetBrains IntelliJ, Apple Xcode
- We will use Visual Studio from Microsoft

Step 4: Implement the Code

- Source code is compiled to check for syntax errors
- Then after several steps, it is translated to machine language
- An executable file (.exe) is created.

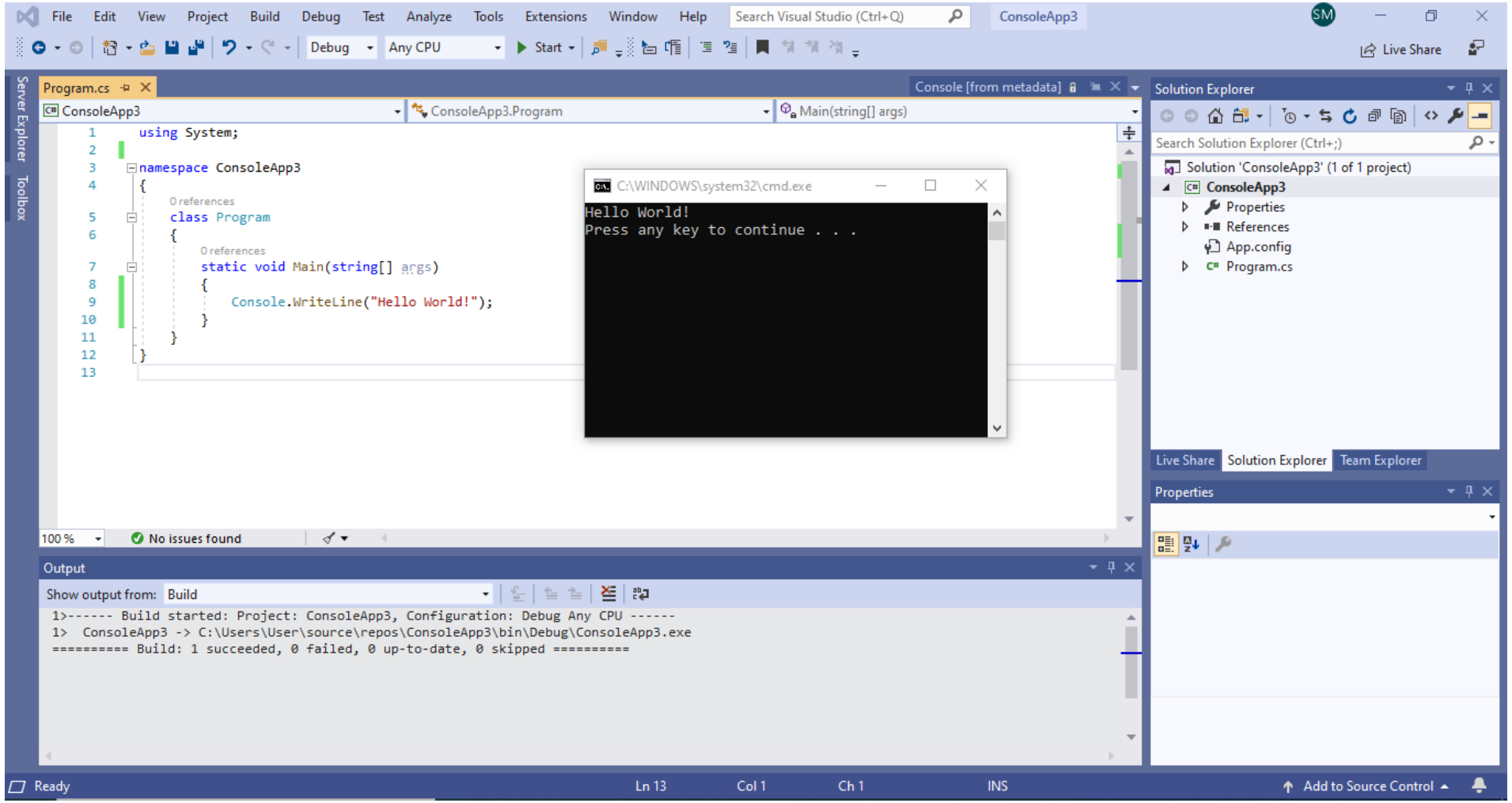
Step 5: Test and Debug

- Test the program to ensure consistent results

3 types of errors:

- Syntax Error
  - Grammar error, Visual Studio catches these errors
- Logic errors
  - Might cause abnormal termination or incorrect results to be produced
- Run-time errors
  - Test plan should include extreme values and possible problem cases

Visual Studio: an IDE



Why C#

- One of the newer programming languages
- Conforms closely to C and C++
- Has the object-oriented class libraries similar to Java

Why C# (continued)

- Can be used to develop a number of applications
  - Software components
  - Mobile applications
  - Dynamic Web pages
  - Database access components
  - Windows desktop applications
- Open source
- C# is object-oriented

Exploring the First C# Program

```
line 1   //  This is traditionally the first program written.
line 2   using System;
line 3   using static System.Console;
line 4
line 5   namespace HelloWorldProgram
line 6   {
line 7      class HelloWorld
line 8      {
line 9         static void Main( )
line 10        {
line 11              WriteLine("Hello World!");
line 12              ReadKey( );
line 13        }
line 14     }
line 15  }
```

Comments in green

Keywords in blue

Elements of a C# Program

- Comments
  - line 1 // This is traditionally the first program written.
  - Like making a note to yourself or readers of your program
  - Not considered instructions to the computer
  - Not checked for rule violations
  - To document what the program statements are doing

Comments

- Comments Benefit: Make the code more readable
- types of commenting syntax
  - Inline comments
  - Multiline comments

Inline Comments

- Indicated by two forward slashes (//)
- Considered a one-line comment
- Everything to the right of the slashes ignored by the compiler
- Carriage return (Enter) ends the comment

```
// This is traditionally the first program written.
```

Multiline Comment

- Forward slash followed by an asterisk (/*) marks the beginning
- Opposite pattern (*/) marks the end
- Also called block comments

```
/*  This is the beginning of a block multiline comment. It can go on for several lines or just be on a
    single line. No additional symbols are needed after the beginning two characters. Notice there is no
    space placed between the two characters. To end the comment, use the following symbols.
*/
```

using Directive

- Permits use of classes found in specific namespaces without having to qualify them
- Framework class library
  - Over 2,000 classes included
- Syntax

```
using namespaceIdentifier;
```

Namespace

- Namespaces provide scope for the names defined within the group
  - Captain example
- Groups semantically related types under a single umbrella
- System: most important and frequently used namespace
- Can define your own namespace
  - Each namespace enclosed in curly braces: { }

Namespace (continued)

- From Example 1-1

Predefined namespace (System)

- line 1 // This is traditionally the first program written.
- line 2 using System;
- line 5 namespace HelloWorldProgram
- line 6 {

User-defined namespace

- line 15 }

Body of user-defined namespace

using Directive

- In addition to referencing namespaces with using directive, can also identify specific static classes
  - New feature available with Visual Studio 2015
    - Cannot specify class name with prior versions
  - Enables you to omit class name when referencing its static class members
- Syntax
  - using static namespaceIdentifier.classname;

Static Class Reference

- From Example 1-1
- line 3 `using static System.Console;`

- line 10 `{`
- line 11 `WriteLine("Hello World!");`
- line 12 `ReadKey( );`
- line 13 `}`

- line 15 `}`

Reference to class members without fully qualifying.

Class Definition

- Building block of object-oriented program
- Everything in C# is designed around a class
- Every program must have at least one class
- Classes define a category, or type, of object
- Every class has a name

Class Definition (continued)

- line 7
- line 8
- line 14

```
class HelloWorld
{

}
```

User-defined class

Class Definition (continued)

- Define class members within curly braces
  - Include data members
    - Stores values associated with the state of the class
  - Include method members
    - Performs some behavior of the class
- Can call predefined classes' methods
  - Main( )

Main( ) Method

- "Entry point" for all applications
  - Where the program begins execution
  - Execution ends after last statement in Main( )
- Can be placed anywhere inside the class definition
- Applications must have *one* Main( ) method
- Begins with uppercase character

Main( ) Method Heading

- line 9 static void Main( )
  - Begins with the keyword static
  - Second keyword (void) is a return type
    - void signifies no value returned
  - Main is the name of the Main( ) method
  - Parentheses "( )" used for arguments
    - No arguments for Main( ) – empty parentheses

Method Body – Statements

- Enclosed in curly braces
  - Example Main( ) method body

```
line 9    static void Main( )
line 10     {
line 11         WriteLine("Hello World!");
line 12         ReadKey( );
line 13     }
```

- Includes program statements
  - Calls to other method
- Here Main( ) calling WriteLine( ) and ReadKey( ) methods

Method Calls

- line 11 WriteLine("Hello World!");

- Program statements
- WriteLine( ) → member of the Console class
- Main( ) invoking WriteLine( ) method
- Member of Console class
- Method call ends in semicolon

Program Statements

- Write ( ) → Member of Console class
  - Argument(s) enclosed in double quotes inside ( )
  - "Hello World!" is the method's argument
  - "Hello World!" is string argument
    - String of characters
- May be called with or without arguments
  - WriteLine( );
  - WriteLine("WriteLine( ) is a method.");
  - Write("Main( ) is a method.");

Program Statements (continued)

- Read( ) and ReadKey( ) accept one character from the input device
- ReadLine( ) accepts string of characters
  - Until the enter key is pressed
- Write( ) does not automatically advance to next line
- Write("An example\n");
  - Same as WriteLine("An example");
  - Includes special escape sequences

C# Elements

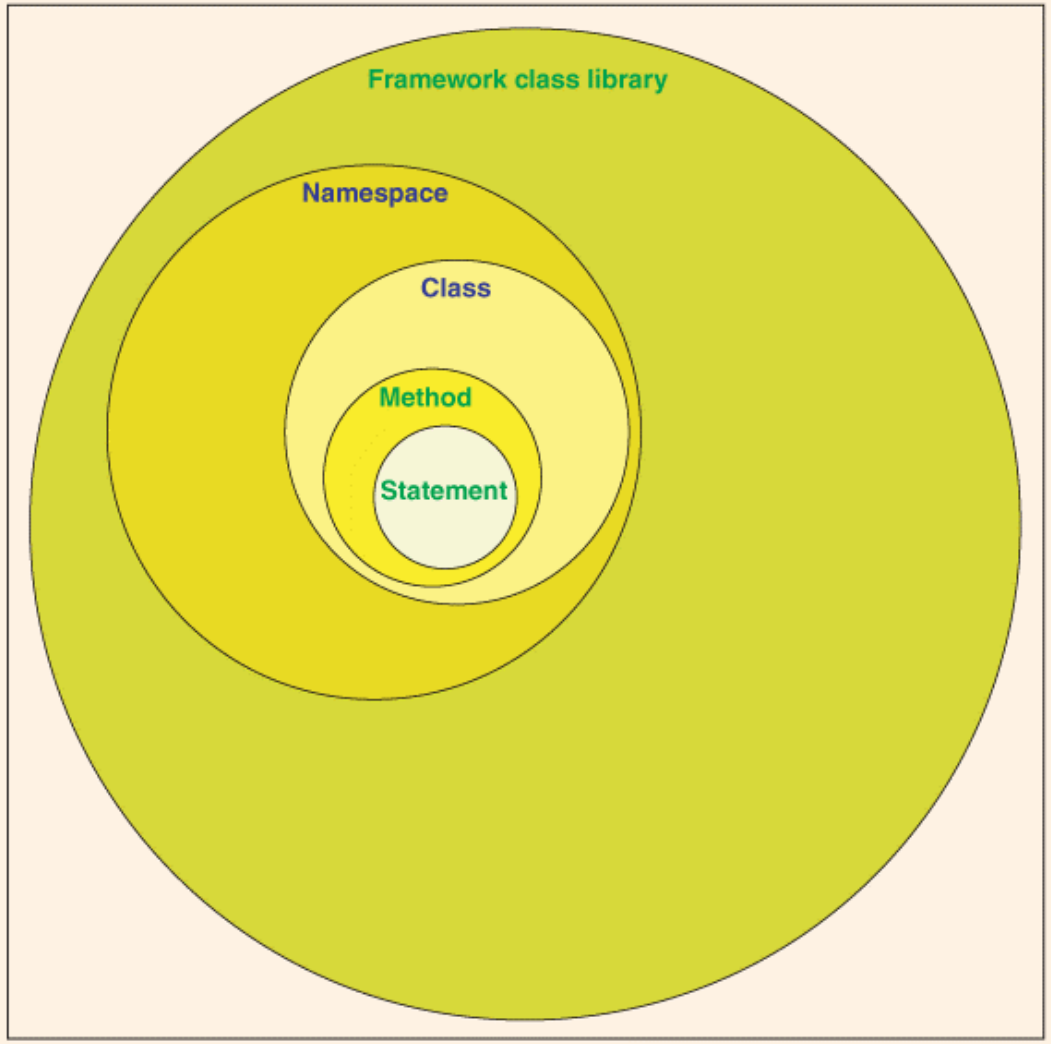


FIGURE 1-15 Relationship among C# elements

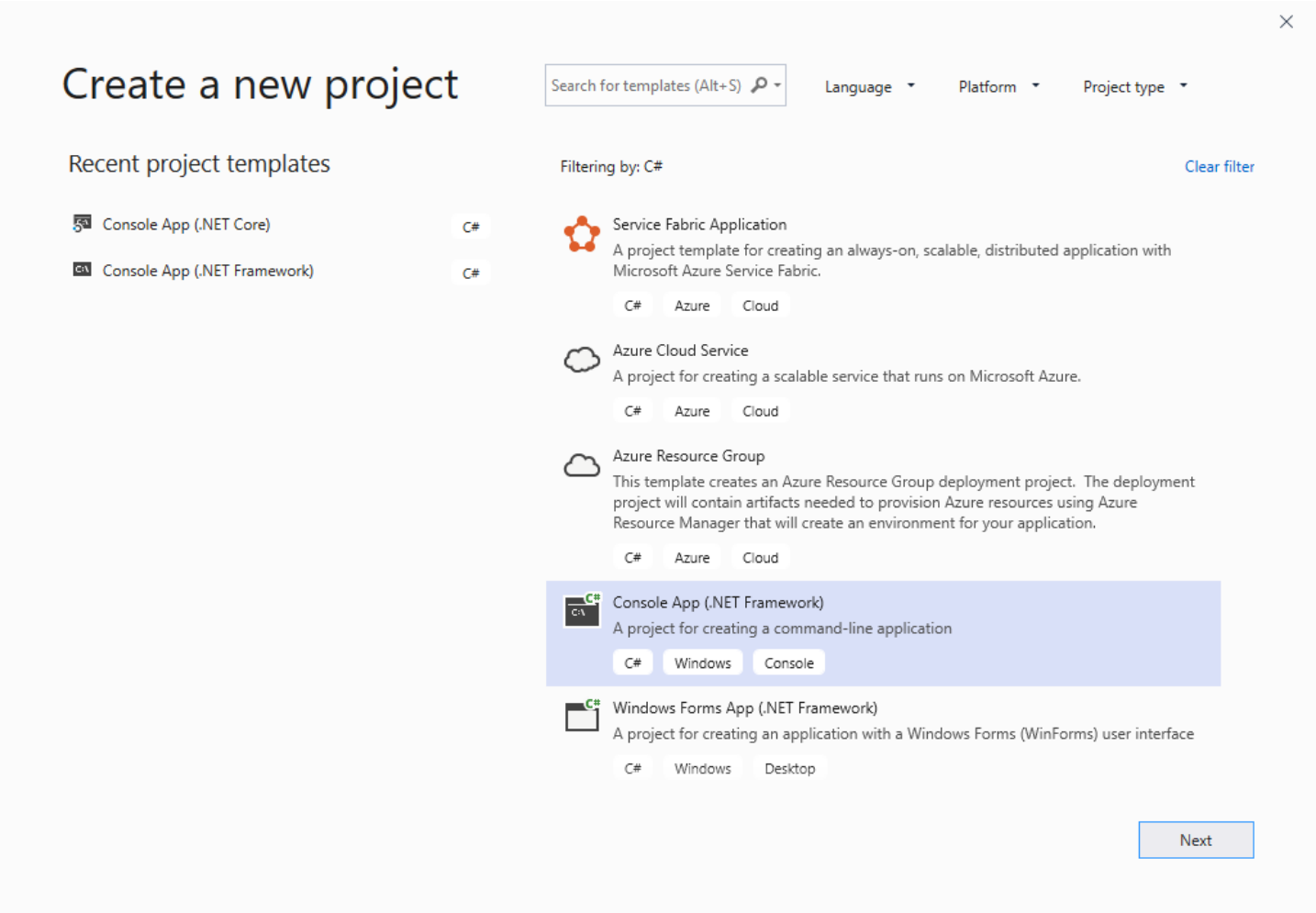
Escape Sequence Characters

Escape sequence character	Description
`\n`	Cursor advances to the next line; similar to pressing the Enter key
`\t`	Cursor advances to the next horizontal tab stop
`\"`	Double quote is printed
`\'`	Single quote is printed
`\\`	Backslash is printed
`\r`	Cursor advances to the beginning of the current line
`\b`	Cursor advances back one position (Backspace)
`\a`	Alert signal (short beep) is sounded

Compiling, Building, and Running an Application

- Begin by opening Visual Studio
- Create new project
  - Select **New Project** on the Start page
  - OR use **File → New, Project** option

Create New Project



Code Automatically Generated

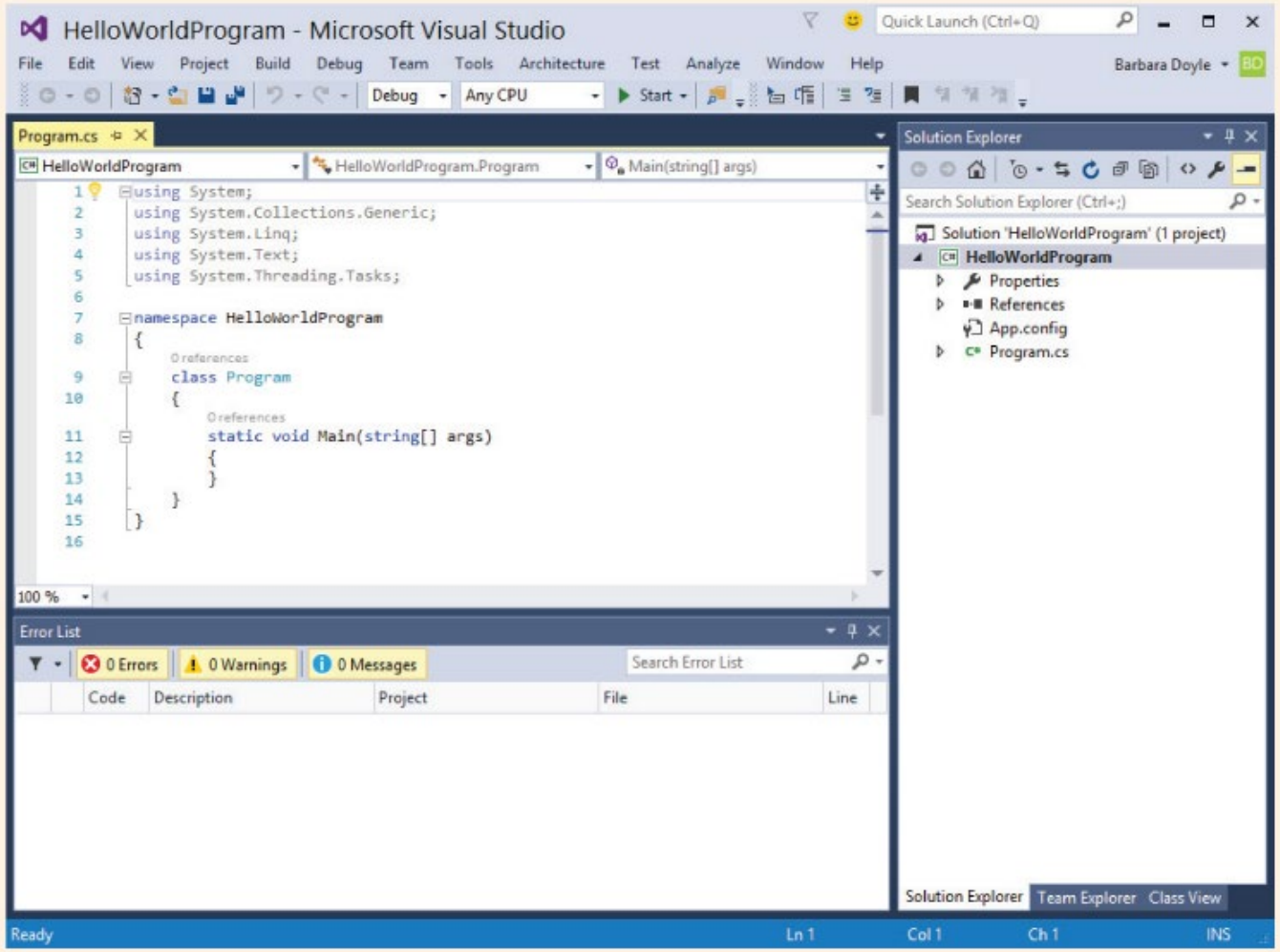
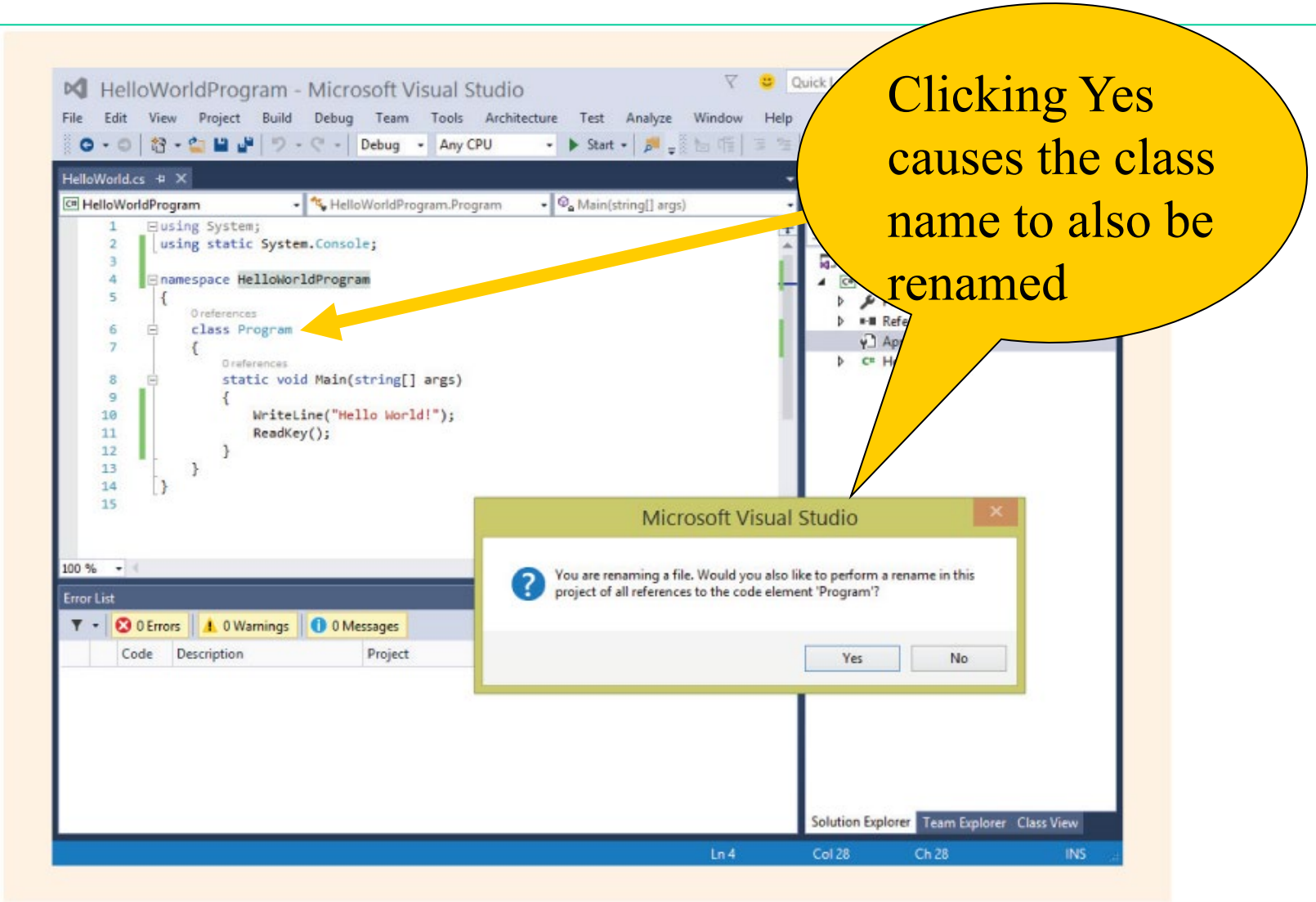


FIGURE 1-17 Code automatically generated by Visual Studio

Typing Your Program Statements

- IntelliSense feature of the IDE
- Change the name of the class and the source code filename
  - Use the **Solution Explorer** Window to change the source code filename
    - Select **View → Solution Explorer**

Rename Source Code Name



Clicking Yes causes the class name to also be renamed

FIGURE 1-18 Changing the source code name from Program

Compile and Run Application

- To Compile – click Build on the Build menu
- To run or execute application – click Start or Start Without Debugging on the Debug menu
  - Shortcut – if executing code that has not been compiled, automatically compiles first
- Start option does not hold output screen → output flashes quickly
  - Last statement in Main( ), could add ReadKey( );

Build Visual Studio Project

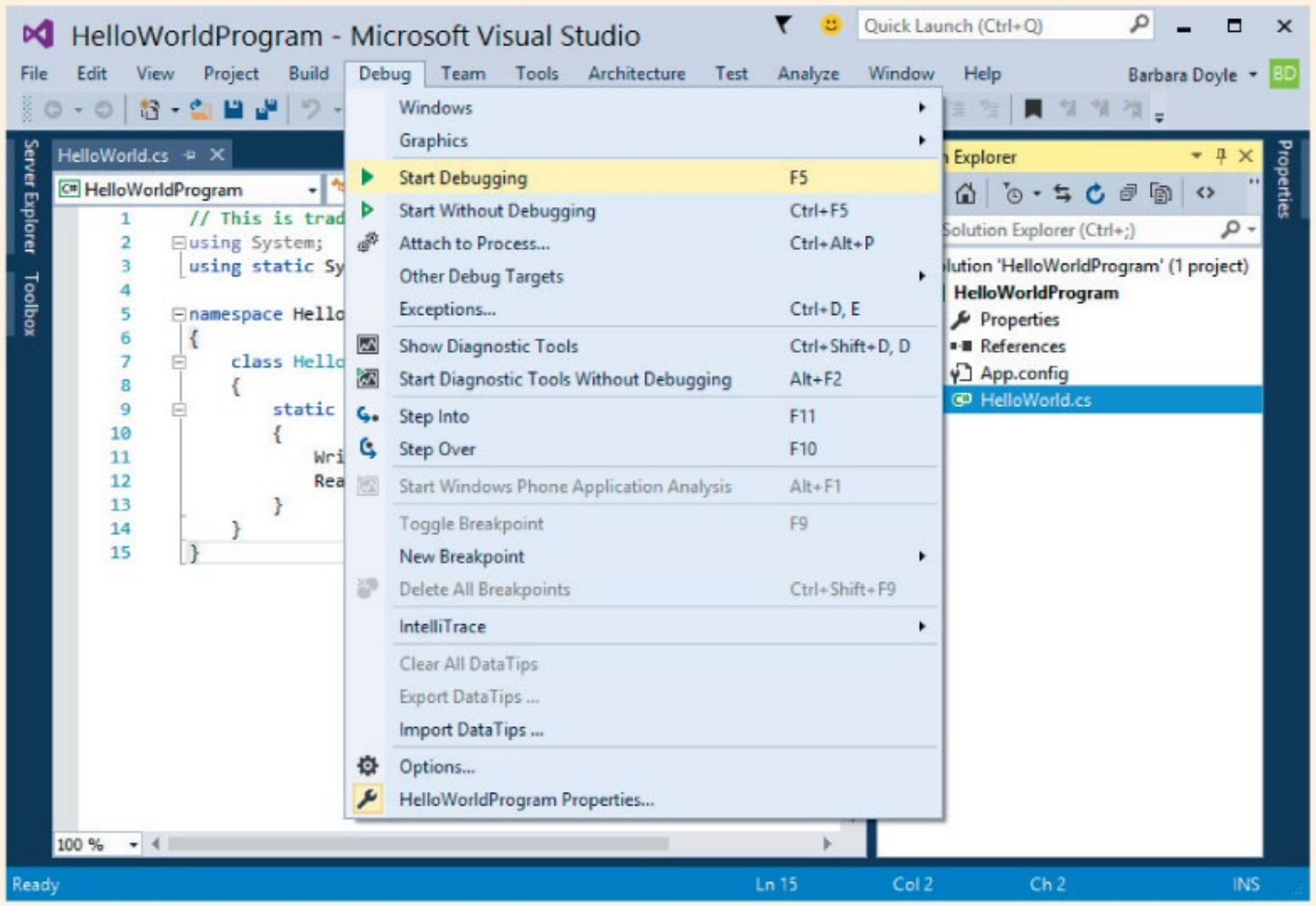


FIGURE 1-19 Execution of an application using Visual Studio

Debugging an Application

- Types of errors
  - Syntax errors
    - Typing error
    - Misspelled name
    - Forget to end a statement with a semicolon
  - Run-time errors
    - Failing to fully understand the problem
    - More difficult to detect

Error Listing

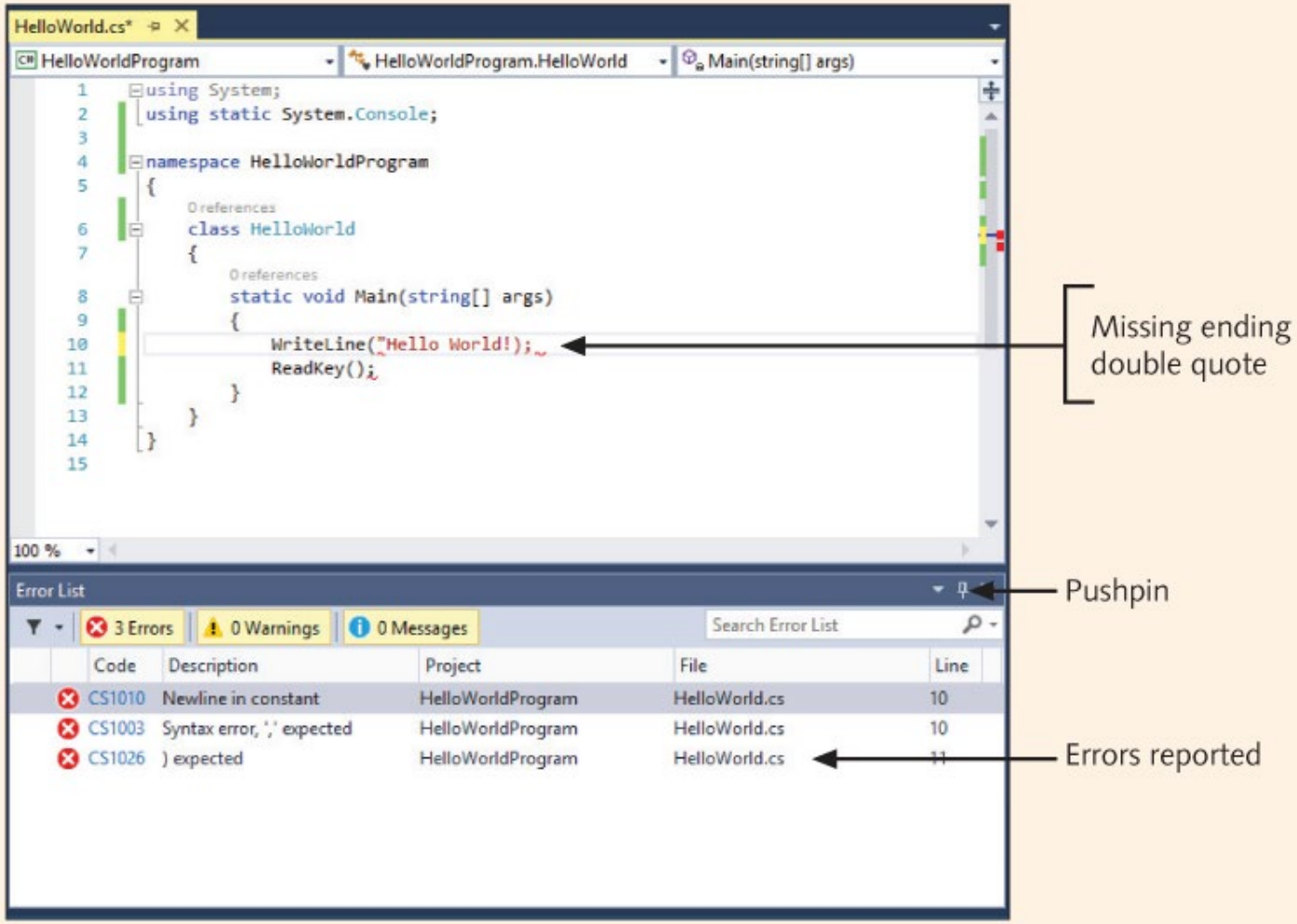


FIGURE 1-20 Syntax error message listing

Creating an Application – ProgrammingMessage Example

PROGRAMMING EXAMPLE: ProgrammingMessage

The problem specification is shown in Figure 1-21.

Create your first C# program.

The purpose of this project is to give you practice in following the steps to software development.

The program should output the following phrase:

Programming can be FUN!

FIGURE 1-21 Problem specification sheet for the ProgrammingMessage example

ProgrammingMessage Example

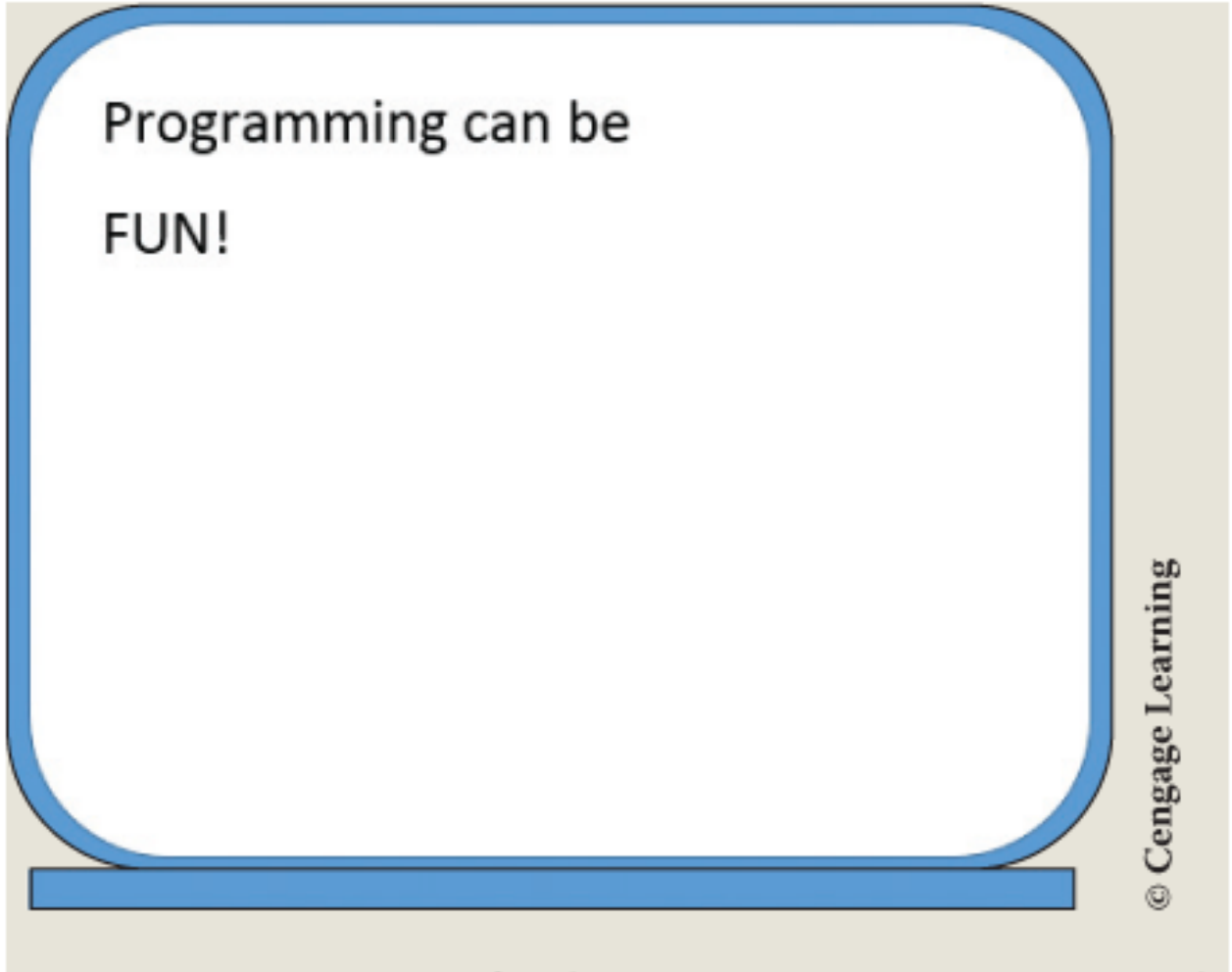


FIGURE 1-22 Prototype for the ProgrammingMessage example

ProgrammingMessage Example

- Pseudocode would include a single line to display the mes
  "Programming can be FUN!" on the output screen

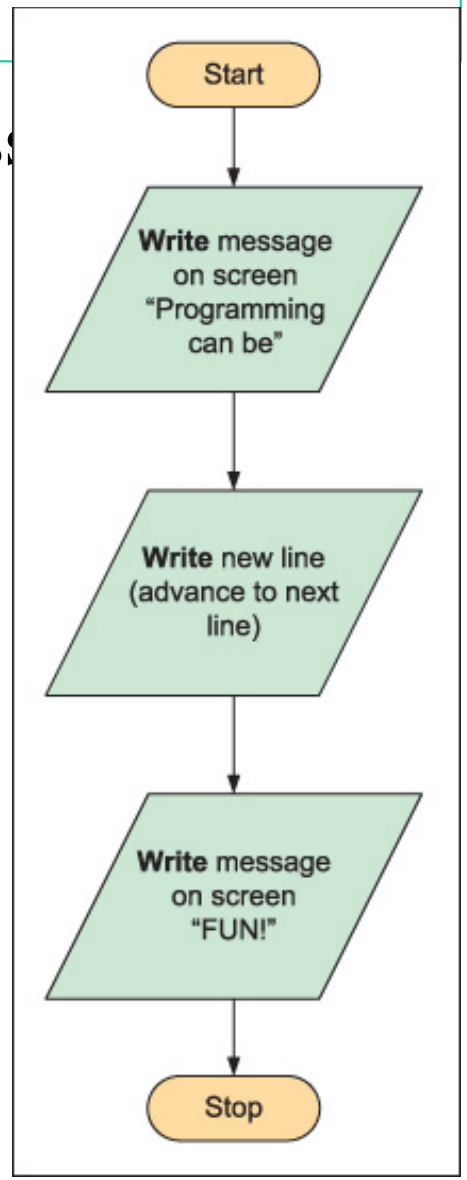


Figure 1-23 Algorithm for ProgrammingMessage example

ProgrammingMessage Example

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ProgrammingMessage
8   {
9       class Program
10      {
11          static void Main(string[] args)
12          {
13          }
14      }
15  }
```

Can delete these using statements

Change the name

Can replace with static void Main()

FIGURE 1-24 Recommended delet

Depending on your current settings, you may not need to make some of these changes

```
/*  Programmer:     [supply your name]
    Date:           [supply the current date]
    Purpose:        This class can be used to send messages to the output screen
*/
using System;
using static System.Console;
namespace ProgrammingMessage
{
   class ProgrammingMessage
   {
            static void Main( )
            {
                 WriteLine("Programming can be");
                 WriteLine("FUN! ");
                  ReadKey( );
            }
   }
}
```

Complete program listing

Coding Standards

- Following standards leads to better solutions
- Following standards makes your code more maintainable
- Following standards saves you time when you go to modify your solution
- Developing standards that you consistently adhere to increases your coding efficiency

Coding Standards - Pseudocode Suggestions

- Use action verbs to imply what type of activities should be performed
- Group items and add indentation to imply they belong together
- Use keywords like while or do while to imply looping
- Use if or if/else for testing the contents of memory locations

Resources

C# Language Specifications –

http://www.microsoft.com/en-us/download/details.aspx?id=7029

C# Programmers Guide –

http://msdn.microsoft.com/en-us/library/67ef8sbd.aspx

History of computing project –

http://www.thocp.net/

Resources

Pascaline –

http://www.thocp.net/hardware/pascaline.htm

The Microsoft .NET Web site –

http://www.microsoft.com/net

The Visual Studio home page –

http://msdn2.microsoft.com/en-us/vcsharp/default.aspx

Resources

U.S. Census Data on Computer and Internet Use –

http://www.census.gov/cps/

Bureau of Labor Statistics Occupational Outlook Handbook –

http://www.bls.gov/ooh/Computer-and-Information-Technology/Software-developers.htm

.NET Foundation for Open Source Development –

http://www.dotnetfoundation.org/

Resources

GitHub Open Source Repository –

https://github.com/Microsoft/dotnet

Mono Cross Platform Open Source Foundation –

http://www.mono-project.com

Xamarin cross-platform mobile development –

https://xamarin.com/

Microsoft Developer Network –

http://msdn.microsoft.com/en-us/

Chapter Summary

- Types of applications developed with C#
  - Web applications
  - Windows graphical user interface (GUI) applications
  - Console-based applications
- Framework class library groups by namespaces
  - Namespaces group classes
  - Classes have methods
  - Methods include program statements

Chapter Summary (continued)

- Programming methodologies
  - Structured procedural
  - Object-oriented
- C#
  - One of the .NET managed programming languages
  - Object-oriented
  - 2001 EMCA standardized
  - Provides rapid GUI development of Visual Basic
  - Provides number crunching power of C++
  - Provides large library of classes similar to Java

Chapter Summary (continued)

- Visual Studio includes .NET Framework
  - Editor tool, compiler, debugger, and executor
  - Compile using Build
  - Run using Start or Start without Debugging
- Debugging
  - Syntax errors
  - Run-time errors
- Use five steps of program development to create applications