Course Learning Outcomes for Unit VII

Upon completion of this unit, students should be able to:

8. Utilize testing methods within a computer program to both verify the program's functionality and also detect and prevent unwanted user input.
   - 8.1 Select appropriate test data for an application.
   - 8.2 Prevent the entry of unwanted characters in a text box.
   - 8.3 Create a message box with the MessageBox.Show method.
   - 8.4 Trim leading and trailing spaces from a string.

Reading Assignment

Chapter 11:
Testing, Testing…1, 2, 3 (Selecting Test Data)

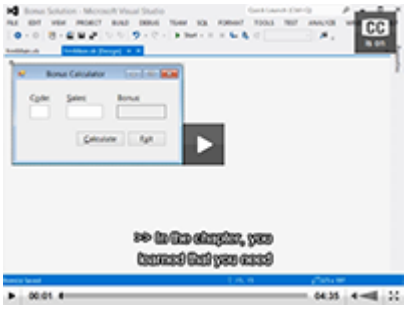
Click here to view the Chapter 11 PowerPoint presentation.

Click here to access a PDF version of this presentation.

Unit Lesson

The last step in the problem-solving process is to rigorously test the program before releasing it to the user. Programs are tested with a set of sample data which includes both valid and invalid data. Message boxes are created to alert the user of important notices. The chapter concludes with a discussion on trimming leading and trailing spaces from strings.

Be sure to use valid and invalid data to test programs.



Click here to view the video.

You can use the testing guidelines illustrated in Figure 1 to guide you in the testing process.

Testing Guidelines

1. Test the application without entering any data.
2. If the application's code expects a text box to contain a number, use both valid and invalid values for the text box. Typically, the test data should include the number 0, positive and negative integers, positive and negative non-integers, and alphanumeric text.
3. If the application's code contains a selection structure, use values that will test each path. If a condition covers a range of values, the test data should include the minimum and maximum values in the range and also a value within the range. If a condition compares strings, include uppercase text and lowercase text in the test data.

Figure 1. Testing Guidelines

The purpose of testing is to try to make the program fail. If you type in the data the program is expecting, you will never find out how the program will handle unexpected input.

When you are satisfied that an application is functioning correctly, you can release it to the user. However, you should keep this fact in mind: no matter how thoroughly you test an application, chances are it still will contain some errors, called *bugs*. Bugs are errors in the code that adversely affect how an application performs. The detection of bugs depends on the efficiency of the testing being done on the software.

In some instances, newly developed applications will go through beta testing. In many organizations, new applications and systems will go through a testing phase before they are released to the customer or to a *live* production network. For example, at one organization, once the applications team completes the development of an application, it would then be tested by the users of the application for several weeks in order to detect any bugs or flaws in the code. As a result of using a formal and structured process such as beta testing, this significantly reduces the failure rate of the software integrations. Figure 2 is an example of a basic software testing process.

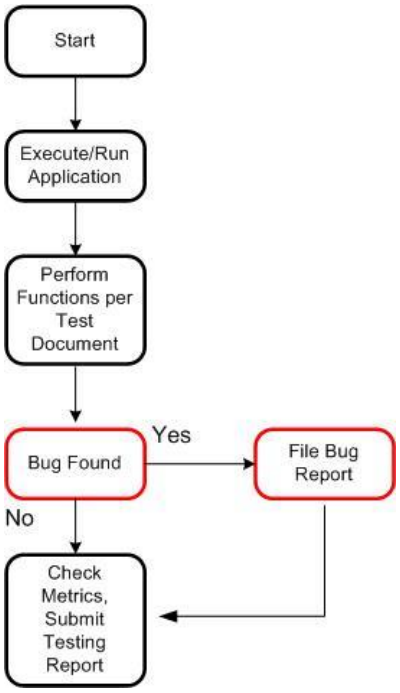


Figure 2. Basic Software Testing Flow

In order to prevent a user from entering an inappropriate character into a text box, you need to use an event procedure called a *KeyPress*. A control's KeyPress occurs each time the user presses a key while the control has the focus. To prevent a text box from accepting an inappropriate character, you first use the *e* parameter's KeyChar property to determine the pressed key.

Suppose you want the user to enter only numbers or numerical data but there are various ways the user can type in the necessary data. For example, say you want the user to type in their birth date in the following format: 00/00/0000. Users may attempt to enter both text and numbers, such as January 10, 1980. This is where the message box comes in handy. When you need to send a message to the user, such as an error message, you send this message using a message box. The message box is represented by the MessageBox.Show method. The syntax in this argument will create a message box and is closed by the user by clicking the *OK* button. Figure 2 illustrates the basic syntax and an example of the message box with an error message for the user. So when the user types in January 10, 1980, you can have a message box pop up informing the user that the input was incorrect and instruct the user in how to correct the input.

When you need to limit the number of characters a user can enter into a text box, you can use the *MaxLength property*. This feature is useful for restricting the amount of text so that when the information is exported to a database where the associated fields may be limited to a certain number of characters. There may be

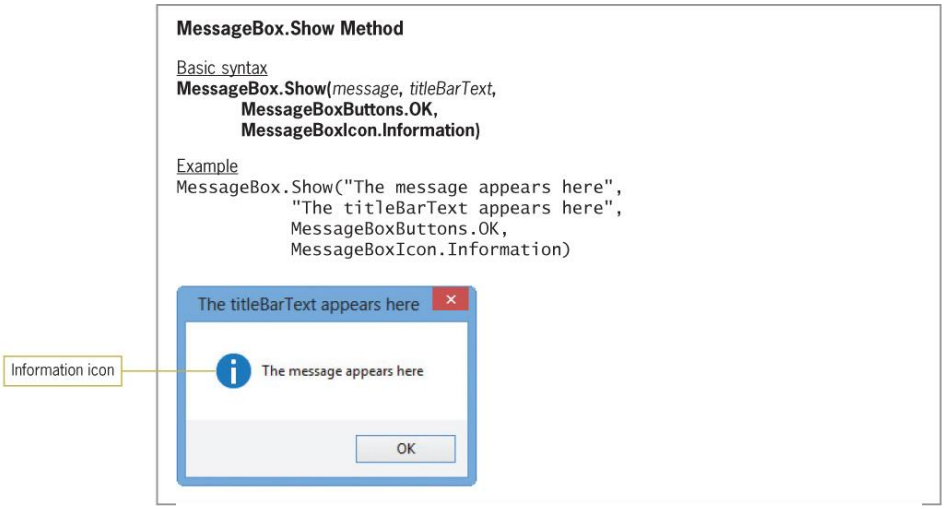


Figure 2. Basic Syntax and Example of the MessageBox. Show Method
(Zak, 2014, p. 252)

instances where *extra* spaces appear in a text box, such as at the beginning and end of a string, but these can be removed using the *Trim method*.

Reference

Zak, D. (2014). *Clearly Visual Basic 2012: Programming with Microsoft Visual Basic 2012* (3rd ed.). Boston, MA: Course Technology.

Learning Activities (Non-Graded)

- Complete Mini Quiz 11-1 on page 245.
- Complete Mini Quiz 11-2 on page 256.
- Complete Exercise 1 (Try This) and Exercise 2 (Try This) on page 259.

Use Appendix B to check your answers.

Non-graded Learning Activities are provided to aid students in their course of study. You do not have to submit them. If you have questions, contact your instructor for further guidance and information.

Key Terms

1. Bugs
2. ControlChars.Back constant
3. Handled property
4. KeyChar property
5. KeyPress event
6. MaxLength property
7. MessageBox.Show method
8. Trim method