

# SOFTWARE TESTING

---

A Craftsman's Approach

---

*THIRD EDITION*

Paul C. Jorgensen



**Auerbach Publications**

Taylor & Francis Group  
Boca Raton New York

---

Auerbach Publications is an imprint of the  
Taylor & Francis Group, an informa business

Visual Basic, Visual FoxPro, and Windows are registered trademarks of Microsoft Corporation.

Java is a trademark of Sun Microsystems, Inc.

ColdFusion is a registered trademark of Adobe Systems Incorporated.

Auerbach Publications  
Taylor & Francis Group  
6000 Broken Sound Parkway NW, Suite 300  
Boca Raton, FL 33487-2742

© 2008 by Taylor & Francis Group, LLC  
Auerbach is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works  
Printed in the United States of America on acid-free paper  
10 9 8 7 6 5

International Standard Book Number-13: 978-0-8493-7475-3 (Hardcover)

This book contains information obtained from authentic and highly regarded sources. Reprinted material is quoted with permission, and sources are indicated. A wide variety of references are listed. Reasonable efforts have been made to publish reliable data and information, but the author and the publisher cannot assume responsibility for the validity of all materials or for the consequences of their use.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access [www.copyright.com](http://www.copyright.com) (<http://www.copyright.com>) or contact the Copyright Clearance Center, Inc. (CCC) 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

**Trademark Notice:** Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

---

**Library of Congress Cataloging-in-Publication Data**

---

Jorgensen, Paul.  
Software testing : a craftsman's approach / Paul C. Jorgensen. -- 3rd ed.  
p. cm.  
Includes bibliographical references and index.  
ISBN-13: 978-0-8493-7475-3 (hardcover : alk. paper)  
ISBN-10: 0-8493-7475-8 (hardcover : alk. paper)  
1. Computer software--Testing. I. Title.

QA76.76.T48J67 2007  
005.1'4--dc22

2007017469

Visit the Taylor & Francis Web site at  
<http://www.taylorandfrancis.com>

and the Auerbach Web site at  
<http://www.auerbach-publications.com>

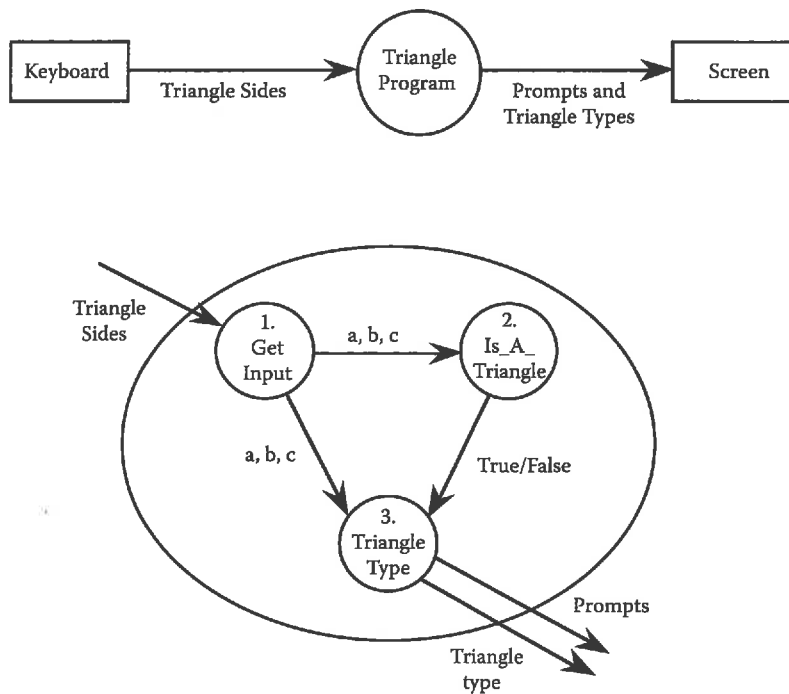


Figure 2.2 Dataflow diagram for a structured triangle program implementation.

Program triangle2 'Structured programming version of simpler specification

```

\
Dim a,b,c As Integer
Dim IsATriangle As Boolean
\
'Step 1: Get Input
Output("Enter 3 integers which are sides of a triangle")
Input(a,b,c)
Output("Side A is ",a)
Output("Side B is ",b)
Output("Side C is ",c)
\
'Step 2: Is A Triangle?
If (a < b + c) AND (b < a + c) AND (c < a + b)
    Then IsATriangle = True
    Else IsATriangle = False
EndIf
\
'Step 3: Determine Triangle Type
If IsATriangle

```

```

Then    If (a = b) AND (b = c)
        Then Output ("Equilateral")
        Else    If (a ≠ b) AND (a ≠ c) AND (b ≠ c)
                Then    Output ("Scalene")
                Else    Output ("Isosceles")
        EndIf
    EndIf
Else    Output ("Not a Triangle")
EndIf
\
End triangle2

Program triangle3 'Structured programming version of improved
specification
\
Dim a,b,c As Integer
Dim c1, c2, c3, IsATriangle As Boolean
\
'Step 1: Get Input
Do
    Output("Enter 3 integers which are sides of a triangle")
    Input(a,b,c)
    c1 = (1 <= a) AND (a <= 200)
    c2 = (1 <= b) AND (b <= 200)
    c3 = (1 <= c) AND (c <= 200)
    If NOT(c1)
        Then Output("Value of a is not in the range of
        permitted values")
    EndIf
    If NOT (c2)
        Then Output("Value of b is not in the range of
        permitted values")
    EndIf
    If NOT(c3)
        Then Output ("Value of c is not in the range of
        permitted values")
    EndIf
Until c1 AND c2 AND c3
Output("Side A is ",a)
Output("Side B is ",b)
Output("Side C is ",c)
\
'Step 2: Is A Triangle?
If (a < (b + c)) AND (b < (a + c)) AND (c < (a + b))
    Then IsATriangle = True
    Else IsATriangle = False
EndIf

```

```

\
Step 3: Determine Triangle Type
If IsATriangle
    Then If (a = b) AND (b = c)
        Then Output ("Equilateral")
        Else If (a ≠ b) AND (a ≠ c) AND (b ≠ c)
            Then Output ("Scalene")
            Else Output ("Isosceles")
        EndIf
    EndIf
Else Output ("Not a Triangle")
EndIf
\
End triangle3

```

## 2.3 The NextDate Function

The complexity in the triangle program is due to relationships between inputs and correct outputs. We will use the NextDate function to illustrate a different kind of complexity — logical relationships among the input variables.

### 2.3.1 Problem Statement

NextDate is a function of three variables: month, day, and year. It returns the date of the day after the input date. The month, day, and year variables have integer values subject to these conditions:

- c1.  $1 \leq \text{month} \leq 12$
- c2.  $1 \leq \text{day} \leq 31$
- c3.  $1812 \leq \text{year} \leq 2012$

As we did with the triangle program, we can make our specification stricter. This entails defining responses for invalid values of the input values for the day, month, and year. We can also define responses for invalid combinations of inputs, such as June 31 of any year. If any of conditions c1, c2, or c3 fails, NextDate produces an output indicating the corresponding variable has an out-of-range value — for example, "Value of month not in the range 1..12." Because numerous invalid day-month-year combinations exist, NextDate collapses these into one message: "Invalid Input Date."

### 2.3.2 Discussion

Two sources of complexity exist in the NextDate function: the complexity of the input domain discussed previously, and the rule that determines when a year is a leap year. A year is 365.2422 days long; therefore, leap years are used for the "extra day" problem. If we declared a leap year every fourth year, a slight error would occur. The Gregorian calendar (after Pope Gregory) resolves this by adjusting leap years on century years. Thus, a year is a leap year if it is divisible by 4, unless it is a century year. Century years are leap years only if they are multiples of 400 (Inglis, 1961), so 1992, 1996, and 2000 are leap years, while the year 1900 is not. The NextDate function also illustrates a sidelight of software testing. Many times, we find examples of Zipf's law, which states that 80% of the activity