Chapter 7 - Methods

Outline

Note: Inconsistent with textbook subsection numbering

[…]

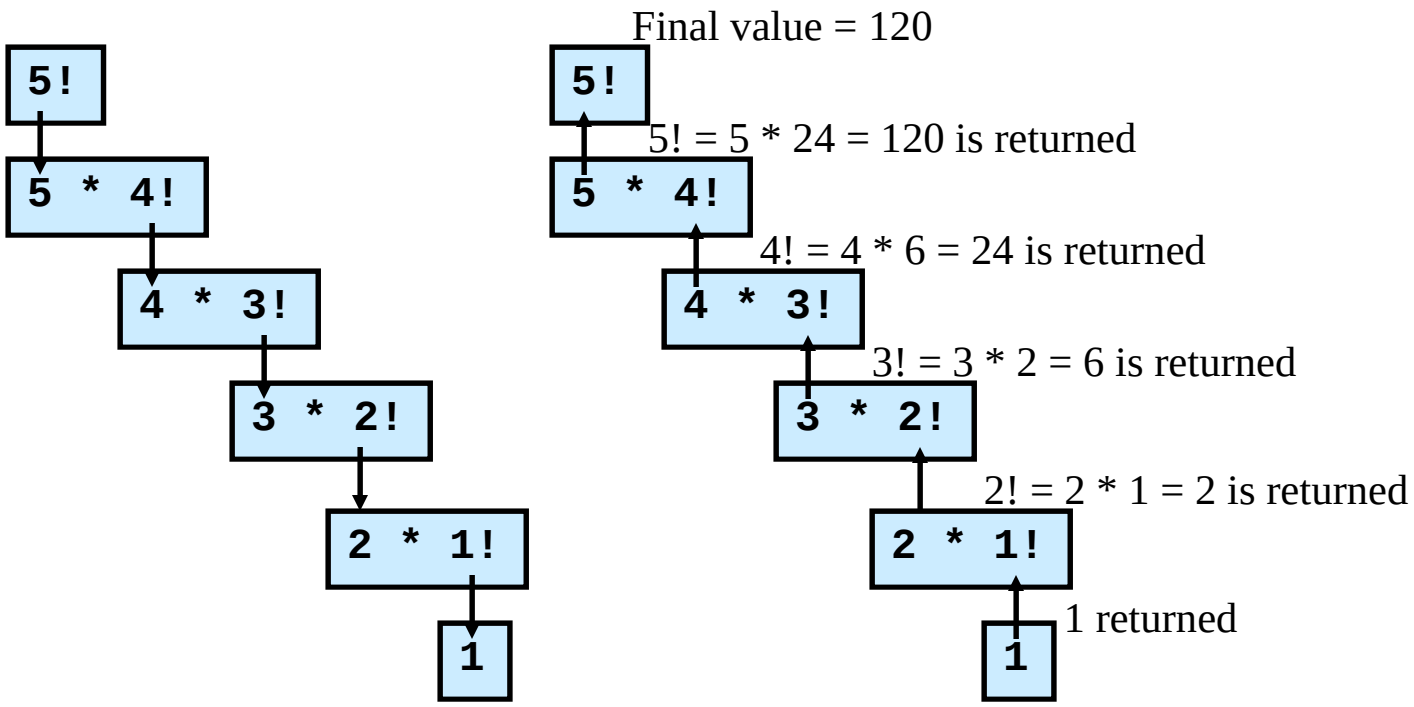
7.13 Recursion

[…]

7.13 Recursion

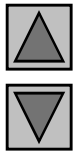
- Recursive methods
  - Methods that call themselves
    - Directly
    - Indirectly
      - Call others methods which call it
  - Continually breaks problem down to simpler forms
  - Must converge in order to end recursion
  - Each method call remains open (unfinished)
    - Finishes each call and then finishes itself

7.13 Recursion



(a) Procession of recursive calls. (b) Values returned from each recursive call.

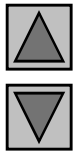
Cf. **Fig. 7.16** Recursive evaluation of 5!.



Outline

FactorialTest.cs

```
1    // Fig. 6.15: FactorialTest.cs [prev. textbook edition]
          // cf. Fig. 7.17 for a simplified version (with console output)
2    // Recursive Factorial method.
3
4    using System;
5    using System.Drawing;
6    using System.Collections;
7    using System.ComponentModel;
8    using System.Windows.Forms;
9    using System.Data;
10
11   public class FactorialTest : System.Windows.Forms.Form
12   {
13      private System.ComponentModel.Container components = null;
14
15      private System.Windows.Forms.Label outputLabel;
16
17      public FactorialTest()
18      {
19         InitializeComponent();
20
21         for ( long i = 0; i <= 10; i++ )
22            outputLabel.Text += i + "! = " +
23               Factorial( i ) + "\n";
24      }
25
```



Outline

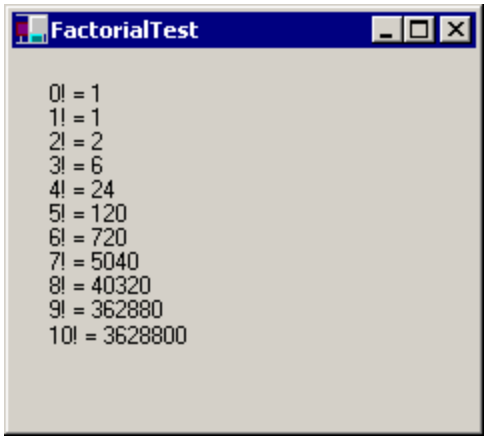
FactorialTest.cs

```
26      // Visual Studio .NET-generated code
27
28      public long Factorial( long number )
29      {
30         if ( number <= 1 )      // base case
31            return 1;
32
33         else
34            return number * Factorial( number - 1 );
35      }
36
37      [STAThread]
38      static void Main()
39      {
40         Application.Run( new FactorialTest());
41      }
42
43   } // end of class FactorialTest
```

The Factorial method calls itself (recursion)

The recursion ends when the value is less than or equal to 1

Program Output



6.15 Example Using Recursion: The Fibonacci Sequence

- Fibonacci Sequence
  - F(0) = 0
  - F(1) = 1
  - F(n) = F(n - 1) + F(n - 2)
  - Recursion is used to evaluate F(n)
- Complexity theory
  - How hard computers need to work to perform algorithms

Recall:

F(0) = 0

F(1) = 1

F(n) = F(n - 1) + F(n - 2)

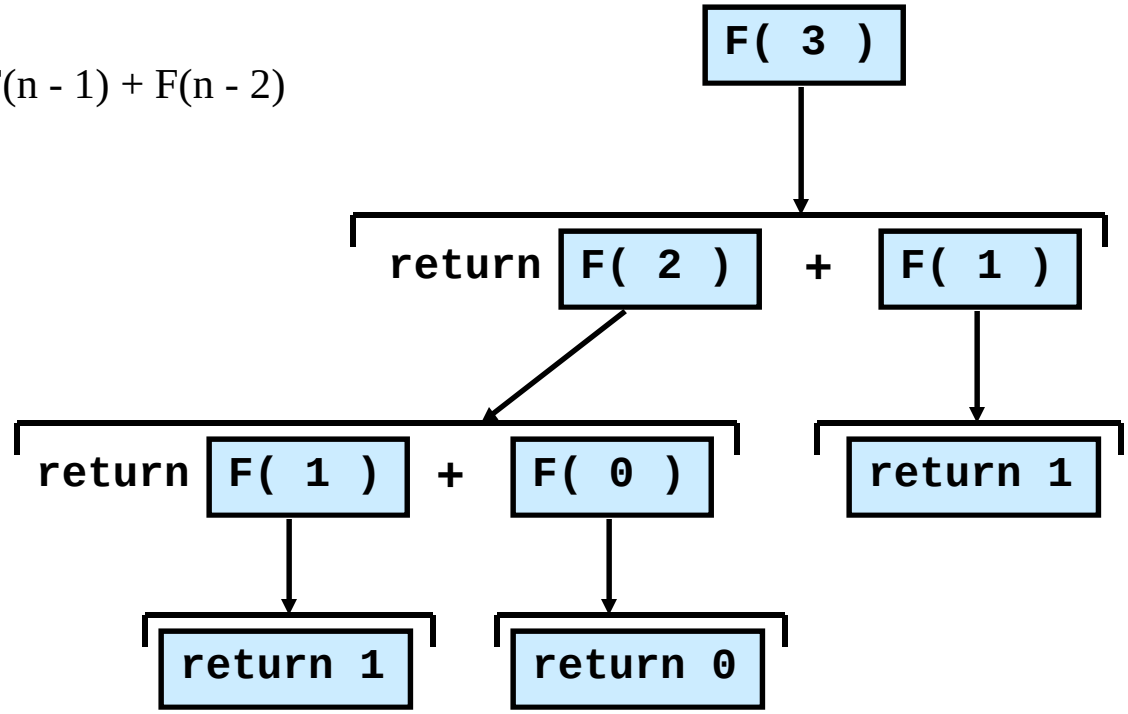
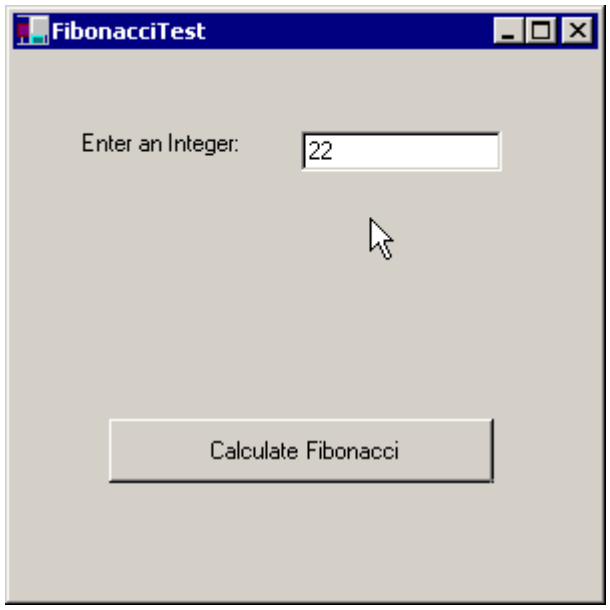


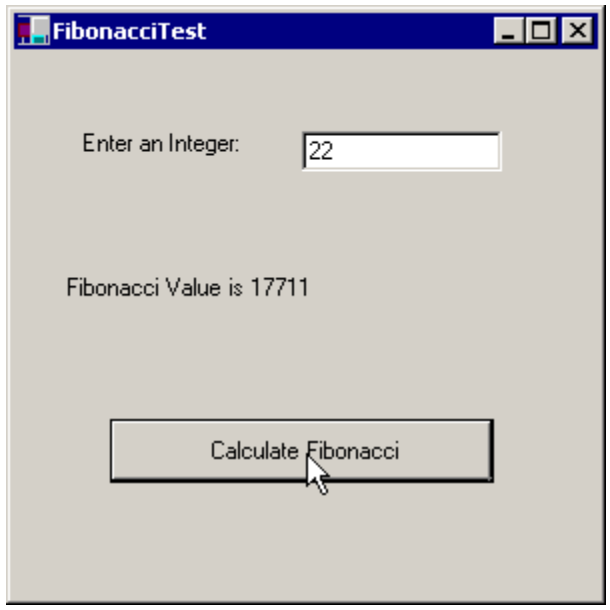
Fig. 6.17 Set of recursive calls to method **`Fibonacci`** (abbreviated as **`F`**).

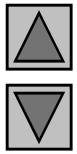
GUI for Recursive Fibonacci Method

Outline

FibonacciTest.cs
Program Output



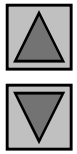




Outline

FibonacciTest.cs

```
1   // Fig. 6.16: FibonacciTest.cs
2   // Recursive fibonacci method.
3
4   using System;
5   using System.Drawing;
6   using System.Collections;
7   using System.ComponentModel;
8   using System.Windows.Forms;
9   using System.Data;
10
11  public class FibonacciTest : System.Windows.Forms.Form
12  {
13     private System.ComponentModel.Container components = null;
14
15     private System.Windows.Forms.Button calculateButton;
16
17     private System.Windows.Forms.TextBox inputTextBox;
18
19     private System.Windows.Forms.Label displayLabel;
20     private System.Windows.Forms.Label promptLabel;
21
22     public FibonacciTest()
23     {
24        InitializeComponent();
25     }
26
27     // Visual Studio .NET-generated code
28
```



Outline

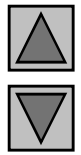
FibonacciTest.cs

```
29      // call Fibonacci and display results
30      protected void calculateButton_Click(
31         object sender, System.EventArgs e )
32      {
33         string numberString = ( inputTextBox.Text );
34         int number = System.Convert.ToInt32( numberString );
35         int fibonacciNumber = Fibonacci( number );
36         displayLabel.Text = "Fibonacci Value is " + fibonacciNumber;
37      }
38
39      // calculates Fibonacci number
40      public int Fibonacci( int number )
41      {
42         if ( number == 0 || number == 1 )
43            return number;
44         else
45            return Fibonacci( number - 1 ) + Fibonacci( number - 2 );
46      }
47
48      [STATh
49      static
50      {
51         Application.Run( new FibonacciTest() );
52      }
53
54   } // end of class FibonacciTest
```

The number uses the Fibonacci method to get its result

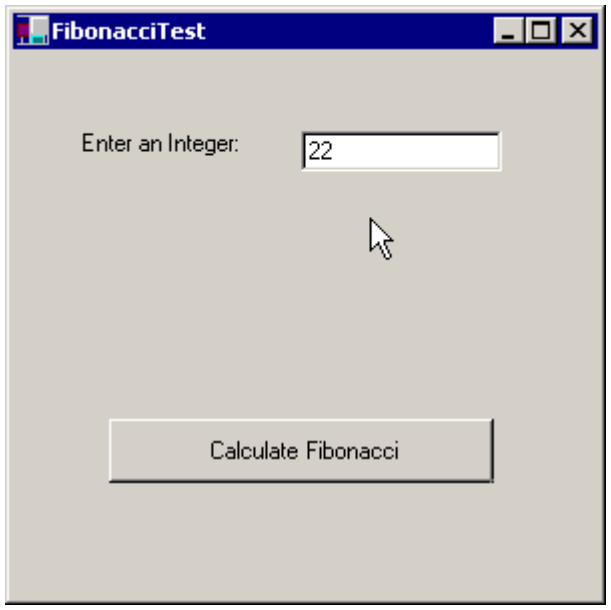
The recursion ends when the number is 0 or 1

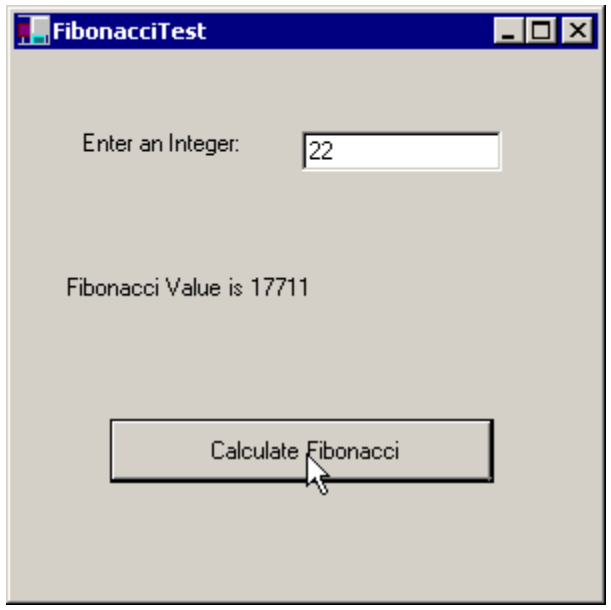
Calls itself twice, to get the result of the the two previous numbers



Outline

FibonacciTest.cs
Program Output





[from previous textbook edition]

6.16 Recursion vs. Iteration

- Iteration
  - Uses repetition structures
    - **`while`**, **`do/while`**, **`for`**, **`foreach`**
  - Continues until counter fails repetition case
- Recursion
  - Uses selection structures
    - **`if`**, **`if/else`**, **`switch`**
  - Repetition through method calls
  - Continues until a base case (e.g., F(0) or F(1)) is reached
  - Creates a duplicate of the variables
    - Can consume memory and processor speed

THE END