Kapitel 8 Beispiel 21

```
 1  private: System::Void bt_rechne_Click(System::Object^  sender,
                                          System::EventArgs^  e)
 2  {
 3     // Flächenberechnung (numerische Integration)
 4     // nach der Simpsonschen Regel
 5     // Autor: Heiderich / Meyer
 6     // ----------------------------------------
 7     // Deklaration der Variablen
 8     int k;          // Iterationsvariable
 9     int n;          // Anzahl Teilintervalle
10     double a;       // untere Intervallgrenze
11     double b;       // obere Intervallgrenze
12     double epsilon; // gewünschte Genauigkeit
13     double delta_x; // Breite der Teilintervalle
14     double xi;      // i. Stützwert
15     double s;       // Summe
16     double aA;      // "alte" Fläche aus Summe
17     double aN;      // "neue" Fläche aus Summe
18     int iaus = 1;   // Steuervariable zur Dokumentation von f(x)
19     int i;          // Zählersteuerungsvariable (for-Schleifen)
20     // Auslesen der TextBoxen
21     a       = System::Convert::ToDouble(this->tB_a->Text);
22     b       = System::Convert::ToDouble(this->tB_b->Text);
23     epsilon = System::Convert::ToDouble(this->tB_epsilon->Text);
24     // Initialisierung "alte Fläche"
25     aA = 0.0;
26     // Zähler für Anzahl Iterationen
27     k = 0;
28     do
29     {
30        // aktueller Zähler für Iterationen hochsetzen
31        k++;
32        // Umspeichern der Ergebnisse
33        aA = aN;
34        // Initialisierung der Summenvariable
35        s = 0.0;
36        // Anzahl Intervalle
37        n = 2* k;
38        // Breite der Teilintervalle
39        delta_x = (b - a) / n;
40        // Berechnung der Summe der Funktionswerte
41        for (i = 0; i <= n; i++)
42        {
43           xi = a + i * delta_x;
44           if (i == 0 || i == n) s += f(xi,iaus);
45           else
46           {
47              if (i / 2 * 2 != i) s += 4.0 * f(xi,iaus);
48              else                s += 2.0 * f(xi,iaus);
49           }
50        }
51        // Berechnung der Fläche aus der Summe
52        aN = delta_x / 3. * s;
53     }while ( abs( aA - aN ) > epsilon);
54     // Ergebnisse anzeigen
55     this->tB_n->Text  = System::Convert::ToString(n);
56     this->tB_aN->Text = System::Convert::ToString(aN);
57  }
58  // Funktion f(x)
59  // Rückgabe Funktionswert an der Stelle x
60  // aus steuert die Dokumentation der Funktion: 0 - keine Ausgabe,
                                                   1 - Ausgabe
61  // Autor: Heiderich / Meyer
62  // Datum: 15.03.2010
63  float f(float x, int& aus)
64  {
65     float y = 4.*sqrt(1-pow(x,2));
66     if (aus == 1)
67     {
68        this->lb_f->Text = "f(x) = 4.0 * sqrt ( 1.0 - x² )";
69        aus = 0;
70     }
71     return y;
72  }
```