
Oldenbourg

C# für Ingenieure

Mit Beispielen zur Analyse elektrischer Schaltungen

von
Lothar Czarnecki

Oldenbourg Verlag München Wien

Bibliografische Information Der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

Rosenheimer Straße 145, D-81671 München
Telefon: (089) 45051-0
www.oldenbourg-verlag.de

Lektorat: Sabine Krüger, Kathrin Veigel
Herstellung: Rainer Hartl
Umschlagkonzeption: Kraxenberger Kommunikationshaus, München
Gedruckt auf säure- und chlorfreiem Papier
Druck: R. Oldenbourg Graphische Betriebe Druckerei GmbH

ISBN 3-486-27357-4

Vorwort

Dieses Buch ist aus den Lehrveranstaltungen für Ingenieure und Wirtschaftsingenieure entstanden, die vom Autor an der Fachhochschule Kempten gehalten werden. Es führt in die Informatik und die Grundgebiete der neuen Programmiersprache C# ein und verwendet dazu Beispiele aus der Elektrotechnik.

Warum ausgerechnet C#? Diese bei der Firma Microsoft entstandene Programmiersprache erfüllt alle Anforderungen, die an eine moderne Sprache gestellt werden müssen: Sie ermöglicht im Gegensatz zu älteren Programmiersprachen wie FORTRAN und C objektorientiertes Programmieren, was für größere Projekte und das Arbeiten im Team unerlässlich ist. Im Gegensatz zu C++ ist sie weitgehend frei von den Fallstricken und Ungereimtheiten, die gerade dem Informatiker im Nebenfach das Leben oft sehr schwer machen. Und schließlich wurde sie von Microsoft, dem größten Softwarehersteller der Welt, auf den Markt gebracht und von ihm selbst für die eigenen Softwareentwicklungen verwendet, was die Garantie gibt, dass sich die Beschäftigung mit ihr nicht irgendwann mangels Verbreitung als nutzlose Investition erweist. Obwohl von Microsoft entwickelt, ist sie überdies standardisiert und somit für andere Anbieter offen.

Ingenieure haben ihre eigene Sicht der Welt. Kein Ingenieur programmiert gerne Beispiele, die mit seinem Fachgebiet nichts zu tun haben – nicht einmal zu Übungszwecken. Daher werden fast ausschließlich Beispiele aus den Ingenieurwissenschaften als Übungsbeispiele verwendet, vorwiegend aus der Elektrotechnik. Im Verlauf des Buchs entstehen Programme, die ausgesprochen nützlich sind – ein Rechner ähnlich demjenigen, der mit dem Betriebssystem Windows als Zubehör mitgeliefert wird, allerdings mit einem großen Unterschied: Dieser Rechner kann auch mit komplexen Zahlen umgehen, die in der Wechselstromtechnik und in der Schwingungslehre eine große Rolle spielen. Außerdem entsteht im letzten Kapitel ein Werkzeug zur Schaltungsanalyse, das ausgezeichnet zur Kontrolle der Lösung von Prüfungsaufgaben aus dem Gebiet der Grundlagen der Elektrotechnik verwendet werden kann.

Das Buch setzt im zweiten Teil (ab Kapitel 9) elementare Grundkenntnisse der Elektrotechnik voraus, wie sie in den ersten beiden Semestern eines Hochschul- oder Fachhochschulstudiums vermittelt werden. Diese Grundlagen werden vor ihrer Verwendung hier nur kurz zusammengefasst und zitiert. Die Grundlagen der Informatik dagegen werden ausführlich eingeführt, ebenso wie die Notationsweisen der Informatik und die grundlegenden Gemeinsamkeiten der Programmiersprachen.

Natürlich bin ich für Anregungen offen. Für Fragen oder Kommentare stehe ich unter

Lothar.Czarnecki@fh-kempten.de

zur Verfügung. Die Beispielaufgaben sollen möglichst von jedem Lernenden selbst gelöst werden. Meine Lösungen stehen auf der Homepage des Verlags zur Verfügung.

Ganz herzlich bedanken möchte ich mich bei meinen Fachkollegen von der Fachhochschule Kempten für viele Anregungen und Diskussionen, besonders bei Prof. Dr. Bernhard Neudecker und Dipl.-Ing. Norbert Grotz für das Korrekturlesen des Manuskripts. Meiner Frau Marlis sei Dank für ihre Geduld und ihr Verständnis mir gegenüber beim Schreiben dieses Buchs.

Haldenwang im Allgäu,

Prof. Dr.-Ing. Lothar Czarnecki

Inhalt

1 Einführung

Die Programmiersprache „C#“ wurde von Anders Heijlsberg bei der Firma Microsoft[1] entwickelt. Sie ist Teil eines Rahmenwerks, des „.net-Frameworks“, das darauf ausgerichtet ist, Anwendungen für das Internet zu entwickeln und stellt die Antwort von Microsoft auf die Sprache „Java“ der Firma Sun[2] dar. Beide Sprachen unterstützen im Gegensatz zur Sprache „C++“ die vollständig objektorientierte Programmierweise[3] und ermöglichen eine saubere Trennung der Datentypen[4]. C# vermeidet darüber hinaus im Gegensatz zu C++ und Java typische Fehler, die entstehen, wenn Operatoren verwechselt werden.

Die Namensgebung soll darauf hinweisen, dass C# (englisch c-sharp oder deutsch cis ausgesprochen, wie der Halbton über c in der Musik) der Nachfolger von C bzw. C++ ist. Tatsächlich erinnert die Syntax der Sprache jedoch eher an Java.

Die gesamte Struktur des Rahmenwerks zeigt Abbildung 1. C# ist eine von mehreren Programmiersprachen, die auf einer gemeinsamen Sprachspezifikation („Common Language Specification“, CLS) aufsetzen. Alle diese Sprachen haben einen gemeinsamen Vorrat an Datentypen und Klassen[5]. Dadurch wird der Übergang zwischen den Sprachen sehr einfach. Aus diesem Grund kann man auch Klassen, die in einer dieser Sprache erstellt wurden, in einer anderen Sprache weiterverwenden. Die CLS baut wiederum auf einer Anzahl Klassen auf, die für die Anwendung zur Verfügung stehen. Die gemeinsame Laufzeitumgebung („Common Language Runtime“, CLR) schließlich ist für die Programmausführung zuständig. Dabei wird der Programmcode zunächst in einen Zwischencode („Intermediate Language“, IL) umgewandelt, der erst unmittelbar vor der Programmausführung in den Maschinencode umgewandelt wird, den der im Rechner vorhandenene Mikroprozessor verstehen und ausführen kann.

---

[1] 1975 von Bill Gates und Paul Allen gegründeter amerikanischer Softwarehersteller mit Sitz in Redmond, Washington.

[2] Sun Microsystems, gegründet 1982.

[3] Beim objektorientierten Programmieren besteht ein Programm aus einer Reihe von Objekten, die Nachrichten untereinander austauschen. Eine genaue Besprechung dieses Programmierkonzeptes erfolgt in Kapitel 7.

[4] Datentypen werden in Kapitel 3 behandelt.

[5] Eine Klasse ist eine Art Schablone, aus der die Objekte erzeugt werden. Näheres hierzu siehe Kapitel 7.

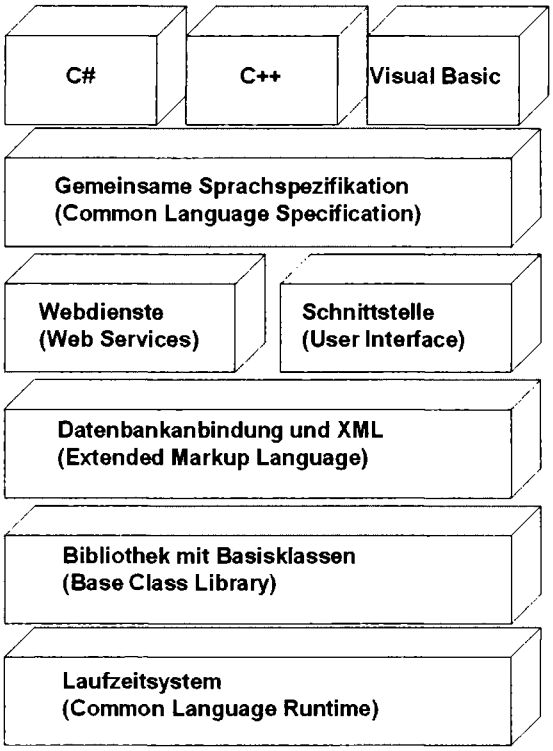

Abbildung 1: Struktur des .net-Frameworks

1.1 Konzeption

Dieses Buch soll den Leser in die Grundkonzepte des modernen objektorientierten Programmierens mit Hilfe dieser neuen Programmiersprache einführen. Dazu werden im nächsten Abschnitt[6] die benötigten Grundlagen aus der praktischen Informatik[7] zusammengestellt. Darauf aufbauend werden in Abschnitt 3 die elementaren Anweisungen, Variablen, Datentypen und Operatoren behandelt. Die Abschnitte 4 und 5 zeigen Kontrollstrukturen im Allge-

[6] Die Begriffe Abschnitt und Kapitel werden in diesem Buch synonym verwendet.

[7] Unter Informatik (engl. computer science) versteht man die Wissenschaft von der systematischen Darstellung, Speicherung, Verarbeitung und Übertragung von Informationen mit Hilfe von Computern. Sie ist unterteilt in die Teilbereiche der theoretischen, praktischen, angewandten und technischen Informatik. Die praktische Informatik beschäftigt sich mit dem Erstellen von Computerprogrammen.

meinen und eine besondere Kontrollstruktur bei der Fehlerbehandlung. Abschnitt 6 beschäftigt sich mit den Prozeduren. Die Abschnitte 7-9 fuhren die objektorientierte Programmierung zunächst allgemein und dann C#-spezifisch ein. Die Abschnitte 10 und 11 behandeln dann zwei spezielle Klassen von C#, nämlich Felder und Zeichenketten. In den übrigen Abschnitten geht es dann um den Entwurf grafischer Oberflächen.

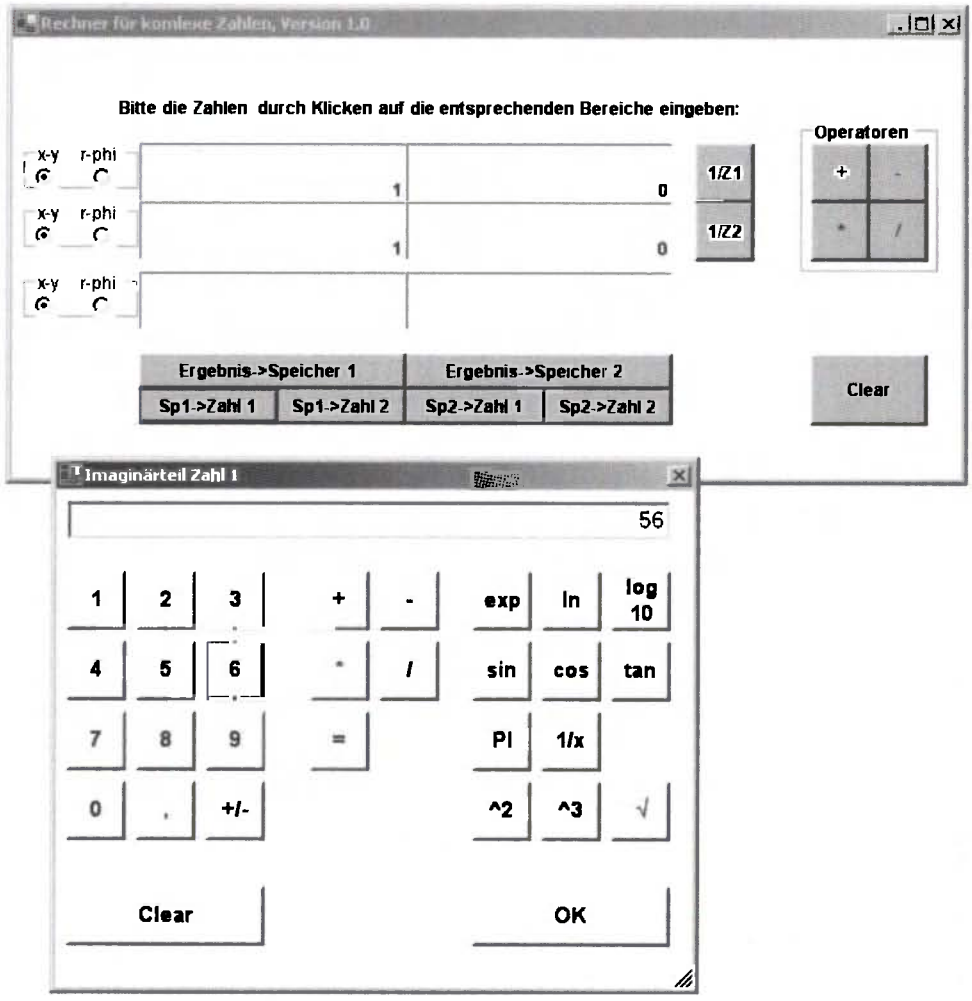

Abbildung 2: Ein "Taschenrechner" mit komplexer Rechnung

Für einen Ingenieur ist eine Programmiersprache nicht Selbstzweck, sondern Werkzeug. Daher entstehen im Rahmen der Beispiele und Übungsaufgaben Programme, die im Lauf des gesamten Elektrotechnikstudiums verwendet werden können. Ein Rechner, der mit komplexen Zahlen umgehen kann, ist hierfür sehr nützlich. Abbildung 2 zeigt die Variante des Autors.

Eines der beliebtesten Prüfungsthemen in den Grundlagen der Elektrotechnik ist die Schaltungsanalyse. Kein Elektrotechnikstudent verlässt die Hochschule ohne gründliche Übung auf diesem Gebiet. Hier kann es sehr hilfreich sein, die dazu notwendigen Lösungsschritte auf den Rechner zu bringen. In den Beispielaufgaben entsteht ein Werkzeug zur Schaltungsanalyse, die Lösung des Autors zeigt Abbildung 3.

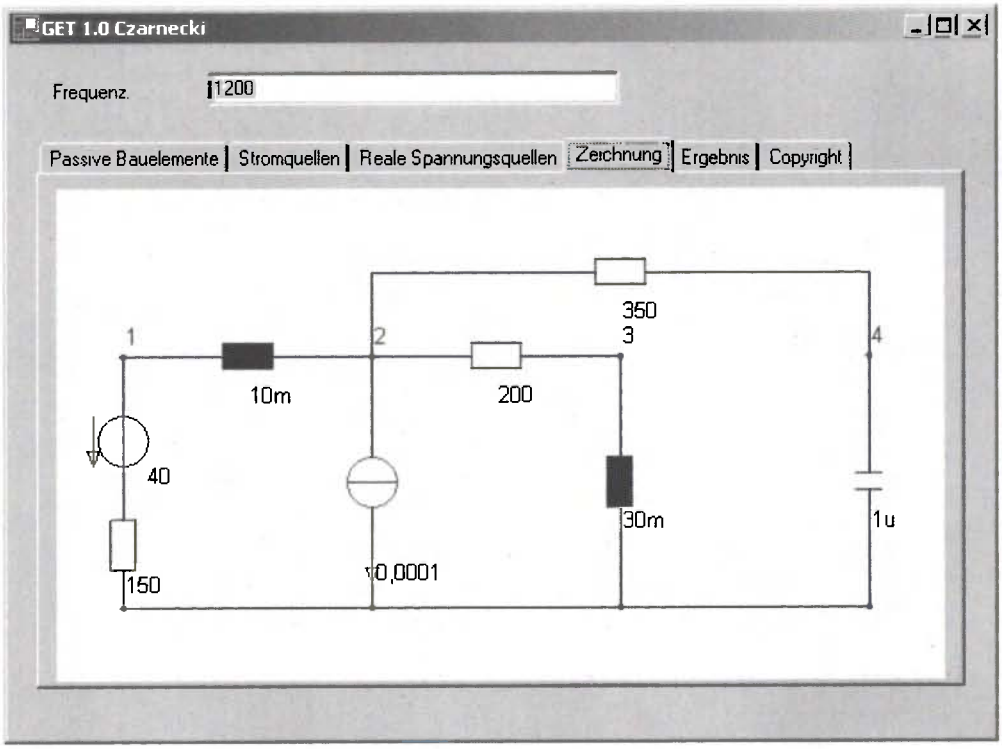

Abbildung 3: Das selbst entwickelte Werkzeug zur Schaltungsanalyse

Als Entwicklungsumgebung wird „Visual Studio.net" von Microsoft verwendet. Diese Entwicklungsumgebung enthält eine Vielzahl von Werkzeugen, die den Programmierer beim Erstellen und Testen unterstützen. Beispielsweise gibt es eine kontextsensitive Hilfe. Wenn man mehr zu einer Klasse oder einem Schlüsselwort erfahren möchte, genügt es daher, die Schreibmarke (den „Cursor") irgendwo in das Wort zu stellen und die F1-Taste zu drücken.

Dann öffnet sich ein Hilfefenster mit dem Eintrag für dieses Wort. Die Namen aller Eigenschaften eines Objektes erscheinen automatisch, wenn man den Namen des Objektes eintippt und anschließend den Dezimalpunkt eingibt (IntelliSense[8]). Ein integrierter Debugger[9] ermöglicht es, schrittweise durch den Programmcode hindurchzugehen und die Werte aller Variablen in einem Beobachtungsfenster anzuzeigen, auch bei Anwendungen fur das Internet. Eine Projektverwaltung erleichtert die Bearbeitung großer Projekte. Insbesondere lassen sich die grafischen Oberflächen der Programme visuell entwerfen.

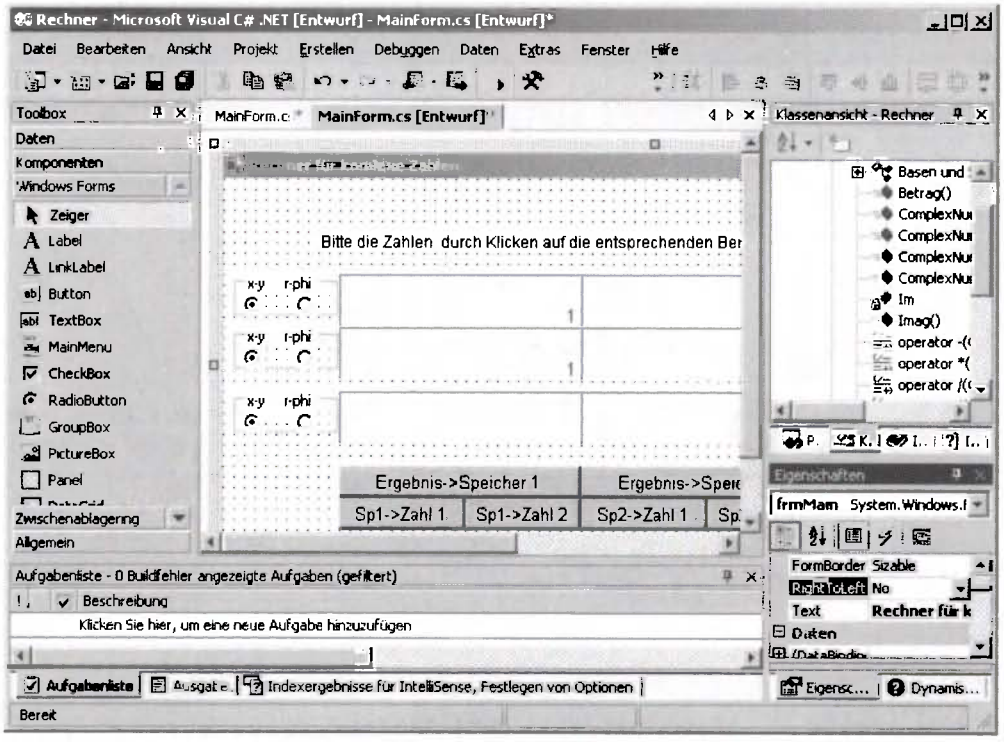

Abbildung 4: Die Entwicklungsumgebung Visual Studio.net der Firma Microsoft

Die Entwicklungsumgebung ist in Abbildung 4 dargestellt. Zu sehen ist das Werkzeug zum visuellen Entwurf einer dialogbasierten Anwendung mit grafischer Oberfläche. Ganz links auf der schmalen vertikalen Leiste sind die Steuerelemente wie Schaltflächen, Textfelder,

[8] Eine Wortschöpfung von Microsoft: Die gewünschten Informationen können im aktuellen Kontext gesucht, Sprachelemente direkt in den Code eingefugt und Eingaben vervollständigt werden.

[9] Das Wort Debugging geht zurück auf den Gebrauch des Wortes bug (dt. Wanze, Insekt, Käfer) fur Fehler in einem technischen System.

Auswahlfelder etc. angeordnet, die mit der Maus auf das Windows-Formular im großen Bereich in der Mitte gezogen werden können. Dabei wird automatisch Programmcode generiert, der anschließend weiter bearbeitet werden kann. Unten am rechten Rand können die Eigenschaften des angewählten Steuerelements verändert werden. Darüber befindet sich eine Ansicht der Klassen, die im Projekt verwendet werden.

Falls Visual Studio.net nicht zur Verfügung steht, kann man die Übungen und Beispiele auch mit kostenlosen Werkzeugen erstellen. Zum Ausführen von bereits fertig entwickelten und übersetzten „.net-Programmen" gibt es das „.net-framework" (Microsoft, 21 MB, kostenlos für Windows 98, ME, NT4, 2000, XP, CE). Für die Entwicklung von C#-Programmen mit der Kommandozeile[10] steht das „.net-framework-SDK" (Microsoft, 131 MB kostenlos, ab NT 4) zur Verfügung. Einen Editor mit Syntax-Highlighting (darunter versteht man das automatische Einfärben von Schlüsselwörtern, Kommentaren, usw.), den „CSharpDevelop-Editor", gibt es kostenlos unter „www.icsharpcode.net".

Microsoft gibt außerdem derzeit[11] auch eine 60-Tage-Testversion von „Visual Studio.net" heraus, die gegen eine Schutzgebühr bezogen werden kann.

1.2 Ein einfaches Beispielprogramm

Ein erstes kleines C#-Programm zeigt das Beispielprogramm 1, welches die Ausgabe eines Textes auf der Befehlszeile und das Einlesen einer Eingabe von dieser Befehlszeile zeigt. Es ist sehr einfach gehalten und läuft nur in einem „MS-DOS-Fenster" („Befehlszeile"). In den neueren Betriebssystemvarianten von Windows heißt dieses Fenster „Eingabeaufforderung". Man erreicht es z.B. im Betriebssystem „Windows 2000" über die „Start-Schaltfläche" links unten (Start->Programme->Zubehör->Eingabeaufforderung).

```
/*
  Hallo.cs:
  Ein erstes Programm mit Ein- und Ausgabe
*/

using System;
class Name
{
  static void Main()
  {
    string Name;
```

[10] Wie man das genau macht, folgt in Kapitel 1.2.

[11] Stand 1.1.2003

```
    Console.Write("Bitte geben Sie Ihren Namen ein: ");
    Name = Console.ReadLine();
    Console.WriteLine("Hallo,  " + Name + "!");
  }
}
```

Beispielprogramm 1: Ein einfaches C#-Programm für die Ein- und Ausgabe auf der Befehlszeile

Wir besprechen kurz den Sinn und Zweck der einzelnen Zeilen: Die erste Zeile ist ein Kommentar, der Informationen für den Leser der Programmzeilen enthält und keinen Einfluss auf die Programmausführung hat. Allgemein gilt: Alles was zwischen /* (Kommentarbeginn) und */ (Kommentarende) steht, wird vom Compiler, der das Programm in Maschinencode übersetzt, ignoriert. Das heißt, dass sich dieser Kommentar auch über mehrere Zeilen erstrecken kann. Einzeilige Kommentare werden dagegen durch zwei Schrägstriche vorwärts (//) eingeleitet. Ein abschließendes Zeichen brauchen wir bei diesem Kommentar nicht, weil das Zeilenende das Kommentarende festlegt.

Im obigen Beispiel ist als Kommentar der Name der Datei angegeben, in der das Programm gespeichert ist sowie eine kurze Erläuterung dessen, was das Programm bewirkt.

Die nächste Zeile (***using System;***) bindet mehrere Klassenbibliotheken ein, die für die Ein- und Ausgabe von Zeichen auf dem Bildschirm benötigt werden. Alle diese Klassen befinden sich in einem „Namensraum" mit dem Namen ***System***. Diese Zeile braucht man praktisch immer. Der Strichpunkt schließt jede Anweisung ab.

Die folgende Zeile, nämlich ***class Name***[12], gefolgt von der öffnenden geschweiften Klammer und der schließenden geschweiften Klammer ganz unten bildet einen „Block", sozusagen eine Umrahmung für den innerhalb stehenden Programmtext. Alles was hier programmiert wird, gehört zu einer Klasse mit dem frei gewählten Namen ***Name***.

In diesem äußeren Block ist ein weiterer Block geschachtelt, in dem das Hauptprogramm steht, welches den festgelegten Namen ***Main()*** trägt und bei dem die Programmausführung beginnt. Innerhalb dieses Blocks wird die eigentliche Funktionalität des Programms festgelegt.

Das Programm verwendet Methoden[13] der Klasse ***Console*** im ***namespace*** („Namensraum") ***System***, nämlich ***Write()***, ***WriteLine()*** und ***ReadLine()***. ***Write()*** und ***WriteLine()*** schreiben eine Ausgabe in ein Fenster mit einer Eingabeaufforderung (auch Befehlszeile oder Konsolenfenster genannt). ***ReadLine()*** liest den eingegebenen Text in das Programm ein.

---

[12] Schlüsselwörter wie ***class*** werden meist klein geschrieben, während Bezeichner wie ***Name*** üblicherweise mit einem Großbuchstaben beginnen. Genaueres hierzu siehe Kapitel 3. Schlüsselwörter und Bezeichner werden im Text fett und kursiv geschrieben, um sie vom übrigen Text abzuheben.

[13] Genaueres zu Methoden finden Sie in Kapitel 7. Wenn Methoden im Text erwähnt werden, werden sie fett und kursiv geschrieben und mit abschließenden runden Klammern versehen.

Um dieses Programm zu übersetzen und auszuführen, können Sie diesen Text zum Beispiel mit dem Editor[14] eingeben, den das Betriebssystem Windows zur Verfugung stellt (unter „Zubehör") und unter dem Namen „Hallo.cs" abspeichern. Achten Sie darauf, dass der Editor nicht automatisch die Dateiendung .txt an den Dateinamen anhängt. Dann suchen Sie mit Hilfe des „Windows-Explorers" das Übersetzungsprogramm (den Compiler) „csc.exe". Falls es nicht vorhanden ist, müssen Sie erst das SDK von Microsoft installieren. Als nächstes öffnen Sie eine „Eingabeaufforderung" und fugen mit dem DOS-Kommando „PATH" dieses Verzeichnis denjenigen Verzeichnissen hinzu, in denen nach Programmen und Dateien gesucht wird, die ohne vollständige Pfadangabe eingegeben wurden. Auf meinem System[15] lautet dieses Kommando:

```
SET PATH=%PATH%;C\:WINDOWS\Microsoft.net\Framework\v1.0.3705
```

Hier dürfen außer zwischen SET und PATH keine Leerzeichen eingegeben werden. Danach geben Sie ***csc***, gefolgt von dem Namen, unter dem Sie die Datei mit dem Programmtext gespeichert haben, ein.

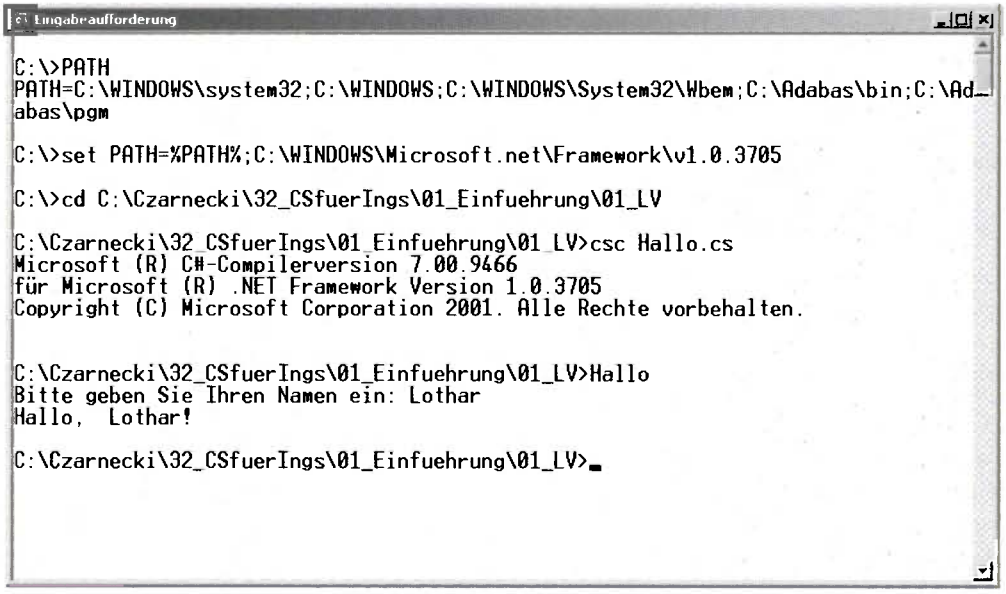

Abbildung 5: Befehlszeile bei Ausführung des Beispielprogramms 1

[14] Mit einem Editor können Texte erstellt und verändert werden.

[15] Windows XP Professional, Visual Studio.net wurde in die standardmäßig vorgeschlagenen Verzeichnisse installiert.

Wenn Sie keine Fehler gemacht haben, entsteht eine ausführbare Datei mit dem Namen „Hallo.exe". Diese kann schließlich durch Eintippen von „Hallo" ausgeführt werden. Den ganzen Vorgang zeigt Abbildung 5.

Diese Möglichkeit zum Übersetzen und Ausführen von C#-Programmen steht jedem zur Verfügung, der ein NT-basiertes Windows-Betriebssystem (NT4, 2000, XP) oder Windows CE auf seinem Rechner hat. Zum Herunterladen der Werkzeuge (SDK, Editor) sollten Sie über einen Internet-Anschluss verfügen.

Wenn Sie die Programmierumgebung Visual Studio.net zur Verfügung haben, klicken Sie auf der Startseite auf die Schaltfläche „Neues Projekt" und wählen in dem dann erscheinenden Dialog als Programmmiersprache „C#" und als Vorlage eine „Konsolenanwendung" aus. Dadurch wird der folgende Programmrahmen automatisch erzeugt:

```
using System;
namespace ConsoleApplication1
{
  /// <summary>
  /// Zusammendfassende Beschreibung für Class1.
  /// </summary>
  class Class1
  {
    /// <summary>
    /// Der Haupteinstiegspunkt für die Anwendung.
    /// </summary>
    [STAThread]
    static void Main(string[] args)
    {
      //
      // TODO: Fügen Sie hier den Code hinzu,
      // um die Anwendung zu starten
      //
    }
  }
}
```

Beispielprogramm 2: Der von Visual Studio erzeugte Programmrahmen

Auch hier findet man wie oben wieder die Prozedur mit dem Namen ***Main()***, allerdings sind die vormals leeren runden Klammern hinter dem Namen ***Main()*** jetzt mit dem Parameterfeld ***string[] args*** gefüllt. Über ihn wird die Übergabe von Parametern beim Programmstart ermöglicht, was aber bei diesem Programm noch nicht notwendig ist. Daher wurde es in Beispielprogramm 1 weggelassen. Wenn man innerhalb der geschweiften Klammern von ***Main()*** die Kommentare durch den obigen Programmcode ersetzt, funktioniert das Programm ganz genauso wie unser erstes. Allerdings brauchen wir noch eine zusätzliche Zeile,

die das automatische Schließen des Konsolenfensters nach der Ausführung des Programms verhindert. Der gesamte Programmtext sieht dann folgendermaßen aus:

```
using System;
namespace ConsoleApplication1
{
  /// <summary>
  /// Zusammendfassende Beschreibung für Class1.
  /// </summary>
  class Class1
  {
    /// <summary>
    /// Der Haupteinstiegspunkt für die Anwendung.
    /// </summary>
    [STAThread]
    static void Main(string[] args)
    {
      string Name;
      Console.Write("Bitte geben Sie Ihren Namen ein: ");
      Name = Console.ReadLine();
      Console.WriteLine("Hallo,  " + Name + "!");
      // Die folgende Zeile ist notwendig,
      // um das Konsolenfenster offenzuhalten.
      // Nach einer beliebigen Tastatureingabe wird
      // das Fenster dann geschlossen.
      Console.Read();
    }
  }
}
```

Beispielprogramm 3: Das zu Beispielprogramm 1 äquivalente Visual-Studio-Programm

Einige Gemeinsamkeiten zu unserem ersten Programm sind leicht zu erkennen. Auch hier werden wieder viele Kommentare verwendet, allerdings diesmal einzeilige (der dritte Schrägstrich hinter manchen Kommentaren wird für eine interne Kennzeichung innerhalb von Visual Studio verwendet). Neu sind der zusätzliche Block, eingeleitet mit dem Wort ***namespace*** und die Wörter vor und nachdem Namen ***Main()***. Auch der Name der Klasse ist hier ein anderer. Diese Unterschiede zum ersten Programm sind im Moment noch nicht wichtig. Das „Attribut" ***[STAThread]*** wurde ebenfalls automatisch hinzugefügt. Es kennzeichnet die Ausführungsweise des Programms. Betrachten Sie das Ganze einfach vorerst als Gerüst, das benötigt wird, um innerhalb von ***Main()*** die eigenen Programmierarbeiten durchzuführen. Die Bedeutung der einzelnen Programmzeilen wird Ihnen nach und nach klar werden, wenn Sie dieses Buch durcharbeiten.
Wenn der Programmcode komplettiert ist, kann das Programm über das Menü von Visual Studio ausgeführt werden. Dazu wählt man zunächst den Menüpunkt „Debug" und anschlie-

ßend „Start“. Dann öffnet sich eine Eingabeaufforderung und das Programm läuft genauso wie in Abbildung 5 ab.

1.3 Übungen

Übung 1:

Geben Sie das angegebene Programm in Ihren Rechner ein, übersetzen Sie es und führen Sie es aus.

2 Grundlagen der Programmierung

In diesem Abschnitt werden die gemeinsamen Grundlagen des Programmierens für alle Programmiersprachen behandelt, soweit sie für unsere Zwecke notwendig sind. Weitergehende Informationen können z.B. aus [1] und [2][16] entnommen werden.

Jedes Fachgebiet hat seine eigene Sprache, seine „Terminologie". Ein zentraler Begriff ist der des „Algorithmus", der sprachunabhängig formuliert werden kann, zur Ausführung auf einem Rechner jedoch in ein Programm, zum Beispiel ein C#-Programm, umgesetzt werden muss. Algorithmen und Terminologie werden in Abschnitt 2.1 behandelt.

Mittel zur Beschreibung der „Syntax" der dazu verwendeten Programmiersprachen sind unter anderem Syntaxdiagramme bzw. die „Backus-Naur-Form", die in Abs. 2.2 erläutert werden.

Bei der Programmentwicklung unterscheidet man verschiedene Entwicklungsphasen. Für die Entwicklung selbst gibt es verschiedene Entwicklungswerkzeuge (Abs. 2.3).

Die Programme werden heutzutage üblicherweise auf einem „Personal Computer" (PC) ausgeführt. Abschnitt 2.4 geht kurz auf Arbeitsweise, Hardware, Software und Speicher ein.

2.1 Terminologie

Unter dem Begriff Softwareentwicklung versteht man die Methoden zur Lösung von Problemen mit dem Computer. Die Erstellung von Computerprogrammen wird als Programmierung bezeichnet. Ein Programm setzt einen Algorithmus, d.h. eine Arbeitsanleitung, in eine computerverständliche Notation um. Beispiele solcher Arbeitsanleitungen aus dem täglichen Leben sind Kochrezepte, Bastelanleitungen, Partituren oder Spielregeln. Dabei ist die Frage, ob diese Arbeitsanleitungen wirklich eindeutig formuliert sind. Eine Partitur zum Beispiel ist es sicher nicht, sie lässt dem Menschen (absichtlich) Freiraum für seine Interpretation.

---

[16] Die Literaturangaben beziehen sich auf die Liste in Abschnitt 16.1.

Alle diese Anleitungen sind ähnlich aufgebaut. Sie bestehen aus Folgen von Anweisungen, die hintereinander ausgeführt werden müssen (z.B. Andante-Allegro-Andante bei der Partitur), aus bedingten Anweisungen (z.B. wenn Geschmack zu fad: Nachsalzen), aus Anweisungsschleifen (rühren, bis die Soße aufgekocht ist) und aus Zutaten oder Voraussetzungen.

Ein Computer kann mit Freiräumen natürlich (zumindest beim gegenwärtigen Entwicklungsstand von Computern) nichts anfangen. Die Arbeitsanleitung zur Lösung eines Problems bzw. einer Aufgabe muss also so präzise formuliert sein, dass sie von einem Computer ausgeführt werden kann. Dafür gibt es Programmiersprachen. Als Hilfe bei der Programmerstellung existieren verschiedene grafische Darstellungen, z.B. „Flussdiagramme" oder „Struktogramme". Die Elemente von Flussdiagrammen und ein Beispiel eines Flussdiagramms zur Bildung der Summe der Zahlen von 1 bis n zeigt Abbildung 6:

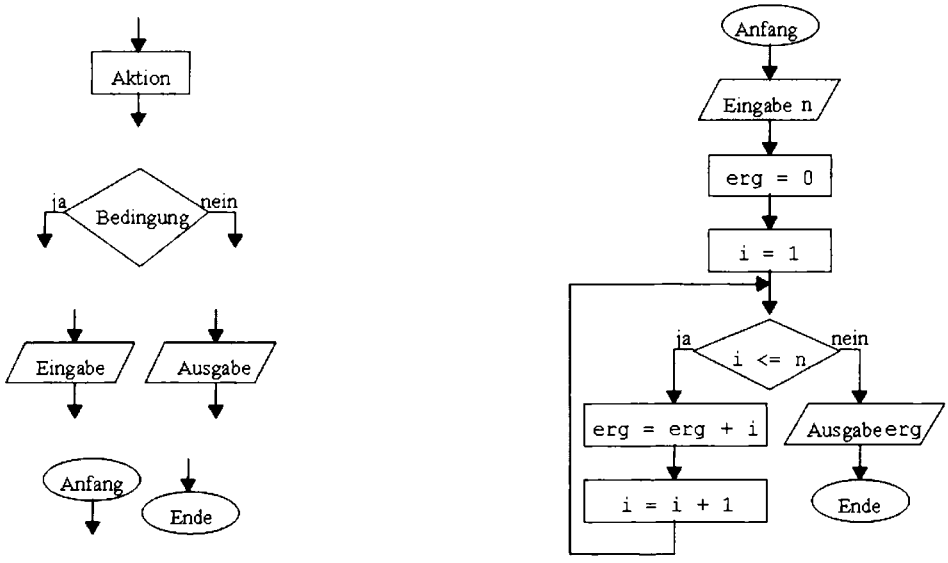

Abbildung 6: Elemente von Flussdiagrammen und ein Flussdiagramm zur Bildung der Summe der ersten n Zahlen.

Das Flussdiagramm symbolisiert die einzelnen Programmschritte als „Programmfluss" von oben nach unten. Verzweigungen werden als Rauten dargestellt. Die Verzweigungsrichtung ist von der Bedingung abhängig, die in die Raute geschrieben wird. Lautet die Antwort auf die als Frage gestellte Bedingung „ja", wird die Programmausführung bei dem mit „ja" beschrifteten Pfeil fortgesetzt, ansonsten bei dem mit „nein" beschrifteten Pfeil. Das Flussdiagramm nach Abbildung 6 enthält eine Schleife, die so lange durchlaufen wird, bis die Bedingung nicht mehr erfüllt ist. Da die Variable ***i*** mit Eins initialisiert wird und sich bei jedem Durchlauf durch die Schleife um Eins erhöht, wird die Schleife ***n*** mal durchlaufen. Bei jedem Durchlauf wird zu dem bisherigen in ***erg*** gespeicherten Wert der Wert ***i*** dazugezählt und

das Ergebnis wieder in ***erg*** gespeichert[17]. Sind alle Schleifendurchläufe abgearbeitet, ist die Summe der Zahlen von 1 bis ***n*** in der Variablen ***erg*** gespeichert und kann ausgegeben werden.

Der in Abbildung 6 beschriebene Algorithmus ist zwar einfach zu verstehen, daher ist er an dieser Stelle auch als Beispiel aufgeführt. Er ist allerdings nicht besonders rationell entworfen. Man könnte das Ergebnis nämlich auch ohne Schleife berechnen. Nehmen wir dazu für ***n*** beispielsweise die Zahl 100 an. Dann gibt der erste Wert, nämlich 1, addiert mit dem letzten, nämlich 100, insgesamt den Wert 101, der zweite Wert 2 mit dem vorletzten Wert 99 ebenfalls usw. Das Ergebnis ist also einfach gleich 50 * 101, und das ergibt die Zahl 5050.

Etwas leichter zu zeichnen als ein Flussdiagramm ist ein Struktogramm, obwohl der wesentliche Inhalt der gleiche ist, wie Abbildung 7 zeigt.

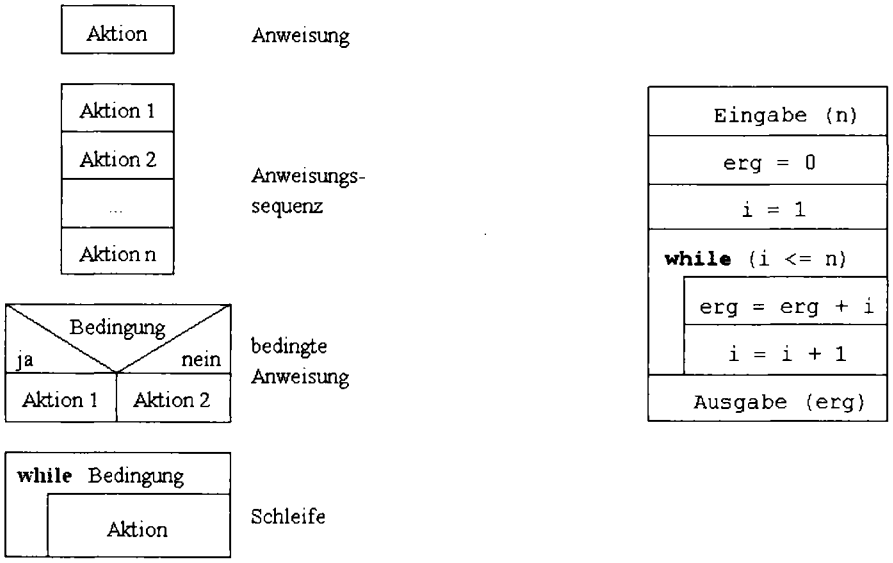

Abbildung 7: Elemente eines Struktogramms und äquivalentes Struktogramm zu Abbildung 6

Eigene Darstellungen für Programmanfang und Programmende existieren hier nicht, ebensowenig für Eingabe und Ausgabe. Dagegen hat die Schleife hier eine eigene Darstellung.

[17] Das Gleichheitszeichen wird in diesem Zusammenhang also anders verwendet als in der Mathematik. Es bedeutet hier folgendes: Werte den Ausdruck rechts vom Gleichheitszeichen aus und speichere das Ergebnis dieser Auswertung in der Speicherstelle, die links vom Gleichheitszeichen steht. Genaueres hierzu siehe Kapitel 3.6.