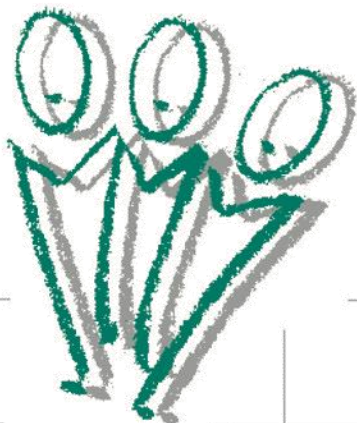


Grundlagen Programmierung

Theorie-Skript Teil 1

Anwendungsentwicklung



Name:
Klasse:

Inhalt

1. Überblick	3
2. Grundlegende Begriffe	4
3. Programmiersprachen und die Entwicklungsumgebung	8
3.1. Programmiersprachen	8
3.2. Entwicklungsumgebung	11
4. Algorithmen und ihre Notationen	13
4.1. Einführung.....	13
4.2. Algorithmus und Programm	13
4.3. Variablen und Arrays	14
4.4. Notationen.....	15
4.5. Kontrollstrukturen.....	16
4.5.1. Folge (Sequenz)	17
4.5.2. Auswahl (Selektion)	17
4.5.3. Wiederholung.....	19
4.5.4. Überblick Kontrollstrukturen	21
4.5.5. Überblick Notationen für Kontrollstrukturen	21
4.5.6. Anwendungsbeispiel Kontrollstrukturen	22
4.5.7. Funktionen als Bausteine (Blockbildung)	25
5. Datenflussplan.....	26

1. Überblick

Das Fach Anwendungsentwicklung spiegelt den Schwerpunkt der Softwareentwicklung in den IT-Berufen wider und behandelt folgende Themenbereiche¹:

1. Analyse, Entwurf, Realisierung und Bereitstellung von Anwendungen in Projekten.
2. Entwicklung und Dokumentation von Programmen auf der Basis grundlegender Algorithmen und Datenstrukturen unter Nutzung einer Softwareentwicklungsumgebung.
3. Entwicklung und Nutzung von Datenbank Anwendungen auf der Grundlage eines Datenmodells unter Anwendung von Datenschutz- und Datensicherungskonzepten.

In der Unterstufe konzentrieren wir uns auf die Erarbeitung der theoretischen Grundlagen zu Punkt 1 und 2, um diese dann in den beiden folgenden Jahren praktisch zu vertiefen. Parallel dazu setzen wir bereits in der Unterstufe Algorithmen in eine Programmiersprache um. Punkt 3 wird, abhängig von der Berufsspezialisierung, in den Mittelstufen und Abschlussklassen behandelt.

Wir beginnen mit der Klärung von Begriffen aus dem Softwarebereich und stellen einige Programmiersprachen und die Entwicklungsumgebung vor.
Danach konzentrieren wir uns auf den Entwurf von Algorithmen.

In den siebziger Jahren wurde aufgrund der immer größer werdenden Datenmengen eine neue Vorgehensweise eingeführt, die so genannte Objektorientierte Programmierung. Im Folgenden erklären wir Notationen und Vorgehensweisen, die diese Programmierung unterstützen.

Wir starten hierbei mit der Erklärung, was Objektorientierte Programmierung ist und welche Vorteile mit ihr verbunden sind.
Danach werden Methoden für die Objektorientierte Analyse und das Objektorientierte Design vorgestellt.

Parallel zu diesem Skript erhalten Sie Übungsmaterial.

¹ Vgl. Lehrplan - Schule in NRW Nr. 4269 für Fachinformatikerin / Fachinformatiker, 1. Auflage 2005
Theorie Skript AW-U FEL/WOR

2. Grundlegende Begriffe

In dem Satz „Analyse, Entwurf, Realisierung und Bereitstellung von Anwendungen in Projekten“ treten eine Vielzahl von wahrscheinlich neuen Begriffen auf, die es zunächst einmal zu klären gilt.

Anwendung

In Wikipedia, dem Projekt zum Aufbau einer freien Enzyklopädie in mehr als 200 Sprachen, wird der Begriff Anwendung wie folgt definiert:

„Ein Anwendungsprogramm (kurz „Anwendung“) ist ein Computerprogramm, das eine für den Anwender nützliche Funktion ausführt, z. B. Buchhaltung, Informationssysteme, Computer Aided Design, Textverarbeitung, Tabellenkalkulation oder auch Spiele. Aus dem englischen Begriff Application hat sich in der Alltagssprache auch die Bezeichnung „Applikation“ für Anwendungsprogramm eingebürgert.

Der Begriff steht im Gegensatz zum Betriebssystem und allen System- und Hilfsprogrammen, die „nur“ den Betrieb ermöglichen, aber noch keinen „Nutzen“ bringen. Auch Programmiersprachen und Werkzeuge zur Softwareerstellung gehören im engeren Sinne nicht zu den Anwendungsprogrammen.“²

Computerprogramm

Dies führt zum nächsten Begriff, dem Computerprogramm:

„Ein Computerprogramm ist eine Folge von Befehlen, die auf einem Computer zur Ausführung gebracht werden können, um damit eine bestimmte Funktionalität (z. B. Textverarbeitung) zur Verfügung zu stellen. Das Computerprogramm liegt zumeist auf einem Datenträger als ausführbare Programmdatei (im so genannten Maschinencode) vor und wird zur Ausführung in den Arbeitsspeicher des Rechners geladen. Das Programm wird als Abfolge von Maschinen- d.h. Prozessorbefehlen von dem Prozessor des Computers (in diesem Fall meist der CPU) verarbeitet und damit zur Ausführung gebracht. Häufig verwendet man für ein Programm auch den englischen Begriff Software.“³

Software

Die folgende Definition zeigt, dass Software aber ein noch umfassenderer Begriff ist:

- Software (engl., eigentlich "weiche Ware"), Abkürzung SW, Sammelbezeichnung für Programme, die für den Betrieb von Rechnersystemen zur Verfügung stehen, einschließlich der zugehörigen Dokumentation (Brockhaus Enzyklopädie).
- die zum Betrieb einer Datenverarbeitungsanlage erforderlichen nichtapparativen Funktionsbestandteile (Fremdwörter-Duden).
- unter Software subsumiert man die immateriellen Teile, d.h. alle auf einer Datenverarbeitungsanlage einsetzbaren Programme (Lexikon der Informatik und Datenverarbeitung / Schneider 86).
- Menge von Programmen und Daten zusammen mit begleitenden Dokumenten, die für ihre Anwendung notwendig oder hilfreich sind (Ein Begriffssystem für die Softwaretechnik / Hesse et al. 84).

² (Wikipedia (o. J.). <<http://de.wikipedia.org/wiki/Anwendung>>. Rev. 2006-07-12.)

³ (Wikipedia (o. J.). <<http://de.wikipedia.org/wiki/Computerprogramm>>. Rev. 2006-07-12.)

- Computer programs, procedures, rules, and possibly associated documentation and data pertaining to operation of a computer system (IEE Standard Glossary of Software Engineering Technology / ANSI 83).
- bezeichnet alle nichtphysischen Funktionsbestandteile eines Computers bzw. eines jeden technischen Gegenstandes, der mindestens einen Mikroprozessor enthält. Dies umfasst vor allem Computerprogramme sowie die zur Verwendung mit Computerprogrammen bestimmten Daten (Wikipedia (o. J.), <<http://de.wikipedia.org/wiki/Software>>. Rev. 2006-07-12.)

Softwareprodukt

Ein Softwareprodukt ist ein Produkt, das aus Software besteht. Es betrachtet die Software von „außen“, aus Käufer- oder Auftraggebersicht.

Dabei ist ein Produkt ein in sich abgeschlossenes, i. a. für einen Auftraggeber bestimmtes Ergebnis eines erfolgreich durchgeführten Projekts oder Herstellungsprozesses.

Softwaresystem

ist ein System, dessen Elemente aus Software bestehen. Es stellt die innere Sichtweise dar, d. h. so wie ein Entwickler die Software sieht.

Dabei ist ein System ein Ausschnitt aus der realen oder gedanklichen Welt, bestehend aus Gegenständen und darauf vorhandenen Strukturen.

Softwareengineering

Die Softwaretechnik (engl. software engineering) beschäftigt sich mit der Herstellung von Software. Eine Definition von Helmut Balzert beschreibt das Gebiet als „zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen.“

Arten von Software

Systemsoftware

ist für eine spezielle Hardwarefamilie entwickelte Software, die den Betrieb und die Wartung dieser Hardware ermöglicht. (Beispiele: Betriebssystem, Compiler, Kommunikationsprogramme, spezielle Dienstprogramme.)

Individualsoftware

ist von einem Kunden in Auftrag gegebene Software. Sie löst speziell dessen Anforderungen. Ist schon abzusehen, welche Mitarbeiter nach der Fertigstellung das Produkt nutzen, dann sollten diese in den Designprozess (klärt, wie die zukünftige Software aussehen soll) mit einbezogen werden.

Branchensoftware

ist eine Software, die zur Darstellung und Bearbeitung von Daten verschiedener Betriebe und Unternehmen einer Branche dient.

Standardsoftware

ist eine Software, die für den anonymen Markt entwickelt wird. Die Anforderungen an die SW werden z. B. von der Marketingabteilung des Software-Herstellers vorgegeben. Oft werden auch Anforderungen von Großkunden bzw. Marktführern berücksichtigt.

Freeware⁴

bezeichnet Software, die vom Urheber zur kostenlosen Nutzung zur Verfügung gestellt wird. Freeware ist meistens proprietär (das bedeutet, dass der Quelltext nicht zur Verfügung gestellt wird).

Freie Software⁵

ist Software, die für jeden Zweck verwendet, studiert, bearbeitet und in ursprünglicher oder veränderter Form weiterverbreitet werden darf. Das schließt auch die kommerzielle Nutzung ein.

Die Free Software Foundation definiert Software als frei, wenn ihre Lizenz folgende Freiheiten einräumt:

- Das Programm zu jedem Zweck auszuführen.
- Das Programm zu studieren und zu verändern.
- Das Programm zu verbreiten.
- Das Programm zu verbessern und zu verbreiten, um damit einen Nutzen für die Gemeinschaft zu erzeugen.

Open Source⁶

Open Source (zu Deutsch „quelloffen“) bezieht sich darauf, dass der Quellcode eines Programms offenliegt. Nicht nur das Programm wird verbreitet, sondern auch der Quellcode, auf Grundlage dessen das Programm erstellt wurde. Wer den Quellcode kennt, kann ihn auch verändern und neue Programme schreiben. Das Bekanntsein des Quellcodes ist wichtig dafür, dass Software frei verbreitet werden kann. Technisch gesehen liegen Freie Software und Open Source nah beieinander, allerdings werden beide Ausdrücke von unterschiedlichen Denkrichtungen genutzt. Freie Software: eher altruistisch; Open Source: eher pragmatisch.

Shareware⁷

ist eine Vertriebsform von Software, bei der die jeweilige Software vor dem Kauf getestet werden kann. Üblicherweise ist es bei Shareware erlaubt, die Software in unveränderter Form beliebig zu kopieren, jedoch im Gegensatz zu Freeware mit einer Aufforderung, sich nach einem Testzeitraum (üblicherweise 30 Tage) beim Autor kostenpflichtig registrieren zu lassen.

⁴ <http://de.wikipedia.org/wiki/Freeware> (abgerufen am 1.4.2010)

⁵ http://de.wikipedia.org/wiki/Freie_Software (abgerufen am 1.4.2010)

⁶ http://de.wikipedia.org/wiki/Freie_Software (abgerufen am 1.4.2010)

⁷ <http://de.wikipedia.org/wiki/Shareware> (abgerufen am 1.4.2010)

Charakteristika von Software

Software ist immateriell, man kann sie nicht anfassen.

Software unterliegt keinem Verschleiß, sie kann beliebig oft ablaufen.

Software altert, da sich die Umgebung, in der die Software eingesetzt wird, ständig ändert.

Wird die Software nicht mit der Änderung der Umgebung angepasst, kann sie irgendwann ihre Aufgabe nicht mehr erfüllen.

Software ist ein umfassenderer Begriff als Programm. Zu einer Software gehört immer eine entsprechende Dokumentation.

Zusammenhänge zwischen Hardware, Daten und Software

Hardware

ist die Gesamtheit aller technischen Geräte eines Rechners, den eigentlichen Computer, Bildschirm Tastatur, Maus, externe Geräte, usw..

Daten

sind Informationen, die zum Zweck der Verarbeitung mit einem Computer aufbereitet und codiert sind.

Hardware, Daten und Software

Die Hardware führt die in der Software codierten Anweisungen aus. Daten werden empfangen, verarbeitet und weitergeleitet. Vom Grundprinzip her kann die Hardware nur ganz wenigen elementaren Instruktionen folgen, den so genannten Maschinenbefehlen. Ein Programm ist nichts weiter als eine Folge solcher Maschinenbefehle, die millionenfach pro Sekunde verarbeitet werden können und technisch zweckmäßig organisiert sind.

3. Programmiersprachen und die Entwicklungsumgebung

Die Realisierung eines Algorithmus in Form eines Programmes wird durch die Verwendung von Programmiersprachen erleichtert.

Programmiersprachen lassen sich folgendermaßen unterscheiden:⁸

3.1. Programmiersprachen

Es gibt mehrere Möglichkeiten, „Programmiersprache“ zu definieren:

- 1) Eine Programmiersprache ist eine formale Sprache (= Menge von Wörtern, welche aus einem gegebenen Alphabet gebildet werden können), die zur Erstellung von Verarbeitungsanweisungen für Rechnersysteme verwendet wird. Die Berechnungen in einem Computer können so in einer für den Menschen lesbaren und verständlichen Form notiert werden.
- 2) Bei einer Programmiersprache⁹ handelt es sich um eine künstliche Sprache, die der Kommunikation mit dem Computer dient. Diese erlaubt die Formulierung von Datenstrukturen und Algorithmen in Form von einem Quellcode, der von einem Prozessor verstanden und ausgeführt werden kann.

Assemblersprache

Eine Assemblersprache ist eine spezielle Programmiersprache, welche die Maschinensprache einer spezifischen Prozessorarchitektur in einer für den Menschen lesbaren Form repräsentiert. Jede Computerarchitektur hat daher ihre eigene Assemblersprache.

Folgendes ist bei allen Assemblersprachen vorhanden:

- Daten lesen und schreiben,
- einfache mathematische und logische Operationen
- einfache Programmflusskontrolle
- Ein- und Ausgabe

Beispiele:

Assembler-Linux, Microsoft Macro Assembler (MASM)

Darstellung der Ausgabe
von „Energiesparhaus“
in Assembler-Linux:

;; *Energiesparhaus for the nasm Assembler (Linux)*

SECTION .data

```
msg      db      "Energiesparhaus",0xa ;
len equ  $ - msg
```

SECTION .text

global main

main:

```
mov    eax,4    ; write system call
```

```
mov     ebx,1      ; file (stdout)
```

```
mov     ecx,msg ; string
```

```
mov     edx,len     ; strlen
```

```
int 0x80 ; call kernel
```

```
mov    eax,1    ; exit system call
```

```
mov ebx,0
```

```
int 0x80 ; call kernel
```

⁸ Vgl. <http://de.wikipedia.org>, abgerufen am 14.5.07

⁹ Vgl. Basiswissen IT-Berufe Anwendungsentwicklung S 74, Bleßmann, Büttner, Dax, Stam-Verlag, 2000

Datenbanksprache

Als Datenbanksprache bezeichnet man eine Untermenge der Programmiersprachen, die für den Einsatz in Datenbanksystemen entwickelt wurden. Mit Hilfe der Datenbanksprache kommuniziert ein Benutzer oder ein Anwendungsprogramm mit dem Datenbanksystem. Zum Sprachumfang gehört in der Regel auch die (Datenbank-) Abfragesprache.

Eine gängige Kategorisierung der Datenbanksprachen oder ihrer Elemente sind die Sparten DDL (Data Definition Language), DML (Data Manipulation Language) und DCL (Data Control Language).

DDL: Anlegen von Datenstrukturen, Ändern und Löschen von Datenstrukturen,

DML: Anlegen, Ändern und Löschen von Daten,

DCL: Rechteverwaltung.

Beispiel: SQL („Structured Query Language“)

Darstellung der Ausgabe
von „Energiesparhaus“
in SQL: `SELECT 'Energiesparhaus';`

Auszeichnungssprache

Eine Auszeichnungssprache dient zur Beschreibung der Daten und teilweise des Verfahrens, das zur Bearbeitung dieser Daten nötig ist.

Beispiel: HTML („Hypertext Markup Language“)¹⁰. HTML ist eine textbasierte Auszeichnungssprache zur Darstellung von Inhalten wie Texten, Bildern und Hyperlinks in Dokumenten. HTML-Dokumente sind die Grundlage des World Wide Web und werden von einem Webbrowser dargestellt.

Darstellung der Ausgabe
von „Energiesparhaus“
in HTML:

```
<HTML>
  <HEAD>
    <TITLE> Energiesparhaus </TITLE>
  </HEAD>
  <BODY>
    Energiesparhaus
  </BODY>
</HTML>
```

Skriptsprache

Skriptsprachen sind Programmiersprachen, die vor allem für kleine, überschaubare Programmieraufgaben gedacht sind. Sie verzichten oft auf bestimmte Sprachelemente, deren Nutzen erst bei der Bearbeitung größerer Projekte zum Tragen kommt. So wird etwa in Skriptsprachen auf den Deklarationszwang von Variablen verzichtet – vorteilhaft zur schnellen Erstellung von kleinen Programmen, bei großen hingegen von Nachteil, beispielsweise wegen der fehlenden Überprüfungsmöglichkeit von Tippfehlern in Variablennamen.

Programme, die in Skriptsprachen geschrieben sind, werden auch Skripte genannt. Skripte werden fast ausschließlich in Form von Quelltextdateien ausgeliefert, um so ein einfaches Bearbeiten und Anpassen des Programms zu ermöglichen.

¹⁰ Vgl. http://de.wikipedia.org/wiki/Liste_der_Programmiersprachen, abgerufen am 14.5.07

Erstes Beispiel:

PHP („PHP: Hypertext Preprocessor“) lehnt sich syntaktisch an C und Perl. PHP wird hauptsächlich zur Erstellung dynamischer Webseiten oder Webanwendungen verwendet. PHP ist freie Software und zeichnet sich durch breite Datenbankunterstützung, Internet-Protokolleinbindung sowie die Verfügbarkeit zahlreicher Funktionsbibliotheken aus.

Darstellung der Ausgabe von „Energiesparhaus“ in PHP:

```
<?php
    // Energiesparhaus in PHP
    echo 'Energiesparhaus!';
?>
```

Zweites Beispiel:

JavaScript (kurz JS) wurde ursprünglich für dynamisches HTML in Webbrowsern entwickelt, um Benutzerinteraktionen auszuwerten, Inhalte anzulegen oder zu verändern. JS erweitert damit die Möglichkeiten von HTML und CSS. Heute wird JavaScript auch auf Servern und in Microcontrollern angewendet.

In JavaScript lässt sich je nach Bedarf objektorientiert, prozedural oder funktional programmieren. JavaScript wird vom Interpreter übersetzt. JavaScript hat keinen Bezug zu Java!

Darstellung der Ausgabe von „Energiesparhaus“ in Javaskript:

```
<script>
    alert('Energiesparhaus')
</script>
```

Objektorientierte Programmiersprache

Eine objektorientierte Programmiersprache ist eine Programmiersprache, deren allgemeine Organisationsstruktur die Klasse ist. Eine Klasse beschreibt Eigenschaften (Attribute) und Verhalten/Fähigkeiten (Methoden/Operationen) einer Gruppe gleichartiger Objekte.

Ein Beispiel für eine Klasse ist der Kunde mit den Eigenschaften Kundennummer, Vorname, Nachname und Adresse. Fähigkeiten des Kunden sind „kaufen“ und „bezahlen“. Ein Objekt der Klasse ist beispielsweise K1425 namens Fritz Müller, der in der Winkelgasse 73 in 45127 Essen wohnt. Allein durch seine Zugehörigkeit zur Klasse Kunde hat er automatisch die Fähigkeiten „kaufen“ und „bezahlen“.

Beispiele:

C++, C#, Java

Darstellung der Ausgabe von „Energiesparhaus“ in C#:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ErstesProjekt
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Energiesparhaus");
            Console.ReadKey();
        }
    }
}
```

3.2. Entwicklungsumgebung

IDE

IDE steht für integrierte Entwicklungsumgebung (engl. *integrated development environment*, auch *integrated design environment*). Es handelt sich hierbei um ein Anwendungsprogramm zur Entwicklung von Software.

Eine IDE verfügt in der Regel über folgende Komponenten:

- Texteditor
- Quelltextformatierungsfunktion
- Compiler bzw. Interpreter
- Linker
- Debugger.

Texteditor

Ein **Texteditor** ist ein Computerprogramm zum Erstellen und Bearbeiten von Texten. Diese Texte enthalten das Coding der jeweiligen Programmiersprache

Quelltextformatierung

Als **Quelltextformatierung** bezeichnet man die Formatierung eines Quelltexts nach bestimmten Regeln (beispielsweise: Schlüsselwörter farbig darstellen, Kontrollstrukturen einrücken). Damit soll die Lesbarkeit des Codings erhöht werden.

Compiler/Interpreter

Je nachdem, ob die Übersetzung eines Programms vor oder während der Ausführung erfolgt, unterscheidet man zwischen kompilierenden oder interpretierenden Übersetzungsprogrammen.

Wird ein Programmtext als Ganzes übersetzt, spricht man in Bezug auf den Übersetzungsmechanismus von einem Compiler. Der Compiler selbst ist ein Programm, welches als Dateneingabe den menschenlesbaren Programmtext bekommt und als Datenausgabe den Maschinencode liefert, der direkt vom Prozessor (Recheneinheit eines Computers, der über Software andere Bestandteile steuert) verstanden oder in einer Laufzeitumgebung ausgeführt wird.

Wird ein Programmtext hingegen Schritt für Schritt übersetzt und der jeweils übersetzte Schritt sofort ausgeführt, spricht man von einem Interpreter.

Interpretierte Programme laufen meist langsamer als kompilierte.

Linker

Unter einem Linker oder Binder (auch: "Bindelader") versteht man ein Programm, das einzelne Programmmodule zu einem ausführbaren Programm zusammenstellt (verbindet). Viele Standardfunktionalitäten werden in Form von Funktionen bzw. Modulen (Programmbestandteilen) zur Verfügung gestellt.

Damit dieser Programmcode in einem anderen Programm verwendet werden kann, wird der Code vom Linker zum Hauptprogramm hinzugefügt.

Der Linkvorgang erfolgt nach der Kompilation.

Debugger

Debugger (engl. *bug*) ist ein Werkzeug zum Auffinden und Diagnostizieren von Fehlern in Hardware und Software.

In einigen Entwicklungsumgebungen gibt es noch die weiteren Werkzeuge: Case Tool und Cast Tool.

Case Tool

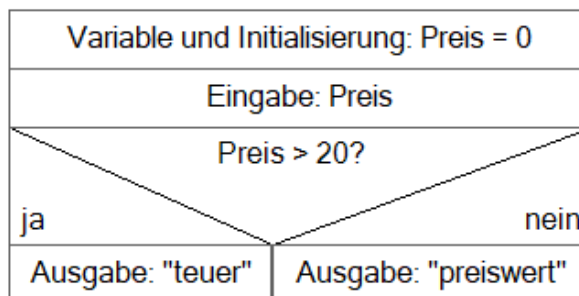
Case Tool steht für Computer-Aided Software Engineering. Es handelt sich um IT- gestützte Werkzeuge zur Umsetzung einer Softwarekonzeption.

CASE-Tools unterstützen grafische Notationen, wie beispielsweise die so genannte UML (Unified Modeling Language) zur Umsetzung der objektorientierten Programmierung oder Datenmodellierungsmethoden, wie das Entity-Relationship-Modell (ERM).

Ebenso kann aus einem Struktogramm-Erstellungs-Tool wie *Struktograf* ein Programmrumpf mit Alternativen- und Schleifen-Syntax in der Programmiersprache C++ entstehen:

Beispiel:

Struktograf



C++-Coding

```
int main( )
{
    // Variable und Initialisierung:
    Preis = 0
    ;

    // Eingabe: Preis
    ;

    //Preis > 20?

    if ( )
    {
        // Ausgabe: "teuer"
        ;

    }
    else
    {
        // Ausgabe: "preiswert"
        ;

    }
}
```

Cast Tool

Cast Tool steht für Computer-Aided Software Testing und bezeichnet Werkzeuge zur Unterstützung von automatisierten Softwaretests.

Beispielsweise kann man mittels eines solchen Tools den Massentest von Kundendaten organisieren. Die Daten ordnet man zunächst in einer sinnvollen Tabellenstruktur und das Tool versorgt dann die Programmabläufe mit den zugeordneten Daten. In diesem Zusammenhang wäre eine Cast-Methode eine Überprüfung, ob die Postleitzahl genau fünfstellig ist, wobei auch eine führende Null akzeptiert wird. Dazu würde man beispielsweise folgende Testdaten zur Verfügung stellen: 03245, -3456, 34875, 232345, 89556

Das Testtool würde den 2. und 4. Testlauf als fehlerhaft anzeigen.

4. Algorithmen und ihre Notationen

4.1. Einführung

Zwischen einer gegebenen Problemstellung und deren Lösung mit Hilfe eines Computersystems besteht in der Regel eine breite Kluft. Zur Umsetzung entwickelt man deshalb so genannte Algorithmen. Das folgende Kapitel definiert den Begriff des Algorithmus und erklärt Grundelemente zur Beschreibung von Algorithmen und deren Notationen. Darüber hinaus wird erläutert, wie ein Algorithmus in ein Programm umgesetzt werden kann. Diese Techniken werden in der Entwurfsphase eines Softwareprojektes angewendet.

Wir konzentrieren uns in diesem Abschnitt auf den 2. Punkt des Lehrplans: „Entwicklung und Dokumentation von Programmen auf der Basis grundlegender Algorithmen und Datenstrukturen unter Nutzung einer Softwareentwicklungsumgebung“.

Wir starten in der Unterstufe mit Entwurfstechniken. Die Umsetzung eines guten Entwurfs in das tatsächliche Programm ist nur ein kleiner Schritt, wenn man die richtigen „Vokabeln“ der Programmiersprache kennt. Hierzu wird das Grundvokabular in der Unterstufe am Beispiel der Programmiersprache C# erlernt und in der Mittelstufe vertieft.

Zunächst starten wir allerdings wieder mit der Klärung der wichtigsten Begriffe!

4.2. Algorithmus und Programm

Algorithmus

Ein Algorithmus ist eine endliche Folge von eindeutig bestimmten Elementaranweisungen, die den Lösungsweg eines Problems exakt und vollständig beschreiben.¹¹

Programm

Ein Programm ist ein Algorithmus, der in einer Sprache formuliert ist, welche die Abarbeitung durch einen Computer ermöglicht.

¹¹ Vgl. Ziegenbal; Algorithmen, Spektrum Verlag Heidelberg, Berlin, Oxford 1999, S. 23 ff.

4.3. Variablen und Arrays

Die Elementaranweisungen eines Algorithmus oder eines Programms benötigen meistens Platzhalter für Werte. Beispielsweise möchte man für einen Artikel einen Nettopreis eingeben und den zugehörigen Bruttowert ausgeben. Programmtechnisch bedeutet dies, dass man Platzhalter für den Artikelnamen, den Nettowert, die Steuer und den Bruttowert benötigt. Diese Platzhalter nennt man Variablen.

In einem Programm erhält eine Variable einen Datentyp. So ist der Artikelname eine Zeichenkette, während der Nettopreis eine Dezimalzahl ist. Die Anzahl der verkauften Artikel wäre eine ganze Zahl (es gibt keine 1,6 Handys).

Beim Anlegen einer Variablen muss der Programmierer den Datentyp (data type) und den Variablennamen (identifier) definieren. Außerdem kann er die Lebensdauer der Variablen beeinflussen. Die Adresse, in der die Variable abgespeichert wird, weist der Rechner automatisch zu.

Als Variablennamen sollten möglichst kurze aber trotzdem sprechende und sinnvolle Namen verwendet werden.

Beispiele:

Nettopreis statt x, falls ein Nettopreis eingelesen wird.

MwSt statt p, falls ein Mehrwertsteuersatz zu 19 % eingelesen wird.

In der Regel dürfen Variablen nicht mit einer Zahl beginnen und sollten keine Sonderzeichen enthalten.

Diese Regeln sollten bereits bei der Erstellung von Algorithmen berücksichtigt werden.

Die Datentypen sind in Abhängigkeit davon auszuwählen, was in der Variablen für Werte gespeichert werden müssen. Die Datentypen heißen in jeder Programmiersprache anders. Die Unterschiede sind allerdings nur gering. Für allgemeine Ausführungen über diese Bestandteile in der jeweiligen Programmiersprache sei auf das IT- Handbuch verwiesen.

Wir beschreiben hier elementare Datentypen für C#:

Datentyp	Beschreibung	Wertebereich
int	Ganze Zahl	-2147483648 bis 2147483747
double	Gleitpunktzahl	-1,7E+308 bis 1,7E+308
float	Gleitpunktzahl	-3,4E+38 bis 3,4E+38
char	ein Zeichen	alle Zeichen
string	Zeichenkette	alle Zeichen
bool	Wahrheitswert	true oder false

Oft hat man mehrere Variablen, die den gleichen Zweck verfolgen, beispielsweise den Umsatz des 1. Quartals, des 2. Quartals, des 3. Quartals und des 4. Quartals.

Es wäre zu umständlich, für jede dieser Information eine separate Variable anzulegen (Quartalsumsatz1, Quartalsumsatz2, Quartalsumsatz3, Quartalsumsatz4).

Stattdessen benutzt man Felder von Variablen (engl. „Arrays“).

Array

Ein Array ist der Sammelbegriff für eine Anordnung von gleichen Elementen in festgelegter Art und Weise.

Mit Hilfe eines Arrays können Daten eines einheitlichen Datentyps (z. B. integer, double, ...) geordnet im Speicher eines Computers abgelegt werden. Der Zugriff erfolgt über einen Index.

Beispiel:

decimal Quartalsumsatz[4]

definiert in C# einen Array mit 4 Elementen. Gezählt wird von Null an, so dass die *Arrayelemente* einen Index von 0 bis 3 erhalten:

Quartalsumsatz[0]

Quartalsumsatz[1]

Quartalsumsatz[2]

Quartalsumsatz[3]

Mit Hilfe der Indexzuordnung können die Felder dann beispielsweise wie folgt gefüllt werden:

Quartalsumsatz [0] = 1000.0

Quartalsumsatz [1] = 2000.0

Quartalsumsatz [2] = 3000.0

Quartalsumsatz [3] = 4000.0

Ebenso können Berechnungen durchgeführt werden, z. B.

Gesamtumsatz = Quartalsumsatz [0] + Quartalsumsatz [1]
+ Quartalsumsatz [2] + Quartalsumsatz [3]

4.4. Notationen

Zum Entwurf und zur Dokumentation von Algorithmen existieren drei Notationen.

Notationen sind standardisierte Formen der Darstellung mittels Grafiken oder speziellen Schlüsselwörtern.

Die drei Notationen sind:


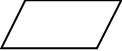

- **Programmablaufplan (PAP) bzw. Flussdiagramme (nach DIN)**
- **Struktogramm (nach DIN)**
- **Pseudocode.**

Alle Notationen gestatten eine übersichtliche und praxisnahe Beschreibung von Algorithmen und sind programmiersprachenunabhängig.

In der Literatur und in den Betrieben sind die Benennungen und Symbole der Kontrollstrukturen nicht einheitlich. Wir verwenden hier als Schulnorm die DIN-Symbole und Benennungen.

Programmablaufplan

Ein Programmablaufplan enthält nach *DIN 66001* genormte Symbole für

Operationen	Ein- und Ausgabe von Daten	Ablauflinien
		

Ein Programmablaufplan ist so aufzubauen, dass er von oben nach unten gelesen werden kann. In die Symbole sollten möglichst knappe, aussagekräftige Texte eingesetzt werden. Der Bearbeitungsanfang und das -ende sind durch das Symbol der Grenzstelle (Termina-

tor¹²⁾ zu kennzeichnen. Wenn der Programmablaufplan zu komplex wird, sollten Konnektoren eingesetzt werden oder der Programmablaufplan ist auf einzelne Programmablaufpläne aufzuteilen. Dabei sollte die Darstellung so einfach wie möglich und so komplex wie nötig sein!¹³⁾

Bei den Programmablaufplänen werden bei bestimmten Kontrollstrukturen zwei verschiedene Darstellungen angegeben. Die erste Darstellung entspricht der Original- DIN-Norm. Die zweite Darstellung wird in der Praxis häufig verwendet.

Struktogramm

Mit Hilfe des Struktogramms kann ein Programm innerhalb eines Strukturblocks dargestellt werden. Häufig werden innerhalb eines solchen Strukturblocks mehrere Unterstrukturblocke geschachtelt. Da nur wenige Basissymbole verwendet werden, ist das Lesen und Modellieren mit Struktogrammen nach *DIN 66261* schnell erlernbar.¹⁴⁾

Pseudocode

Der Pseudocode ist eine textuelle Darstellungsform und weist sehr große Ähnlichkeit mit dem späteren Programm auf. Für den Pseudocode existiert allerdings keine DIN-Norm. In der Literatur findet man unterschiedliche Darstellungen. Es werden deutsche oder englische Schlüsselwörter wie beispielsweise „*wenn-dann-sonst*“ bzw. „*if-then-else*“ verwendet. Logische Trennungen erfolgen durch Einrückungen oder Verwendung von geschweiften Klammern und Semikolon als Befehlsabschluss.

Wir verwenden hier eine Variante, die in der wissenschaftlichen Programmierung benutzt wird. Diese beruht im Wesentlichen auf englischen Schlüsselwörtern und Einrückungen, da dies für den Menschen übersichtlicher und leichter lesbar ist („der Mensch ist kein Compiler“).

4.5. Kontrollstrukturen

Für den zielgerichteten Ablauf eines Algorithmus ist es notwendig, die Reihenfolge der auszuführenden Arbeitsschritte, die so genannten elementaren Aktionen, festzulegen und gegebenenfalls bedingungsabhängig zu steuern. Eine solche Ablaufsteuerung wird als Kontrollstruktur bezeichnet. Sie bestimmt die Reihenfolge, die Auswahl und Häufigkeit der Ausführungen von Aktionen in einem Algorithmus.

Für jede Kontrollstruktur existiert eine von der Programmiersprache abhängige Übersetzung in Programmcode. Dieser Programmcode wird in Maschinencode übersetzt, so dass der Prozessor ihn auswerten kann und das gewünschte Ergebnis liefert.

Man unterscheidet bei der prozeduralen Programmierung folgende Kontrollstrukturen:

- Folge (Sequenz)
- Auswahl (Selektion)
- Wiederholung (Iteration)

In den folgenden Abschnitten werden die einzelnen Kontrollstrukturen für die drei unterschiedlichen Notationen dargestellt.

Hierbei werden in den Symbolen folgende Abkürzungen verwendet:

- G: gemeinsamer Bedingungsteil
- B: Bedingung
- V: Verarbeitung

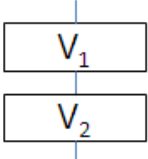
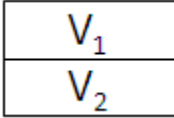
¹²⁾ Vgl. Grenzstellensymbol für Datenflusspläne

¹³⁾ Vgl. Bleßmann, Jörg / Büttner, Artur / Dax, Erwin, Basiswissen IT-Berufe, Anwendungsentwicklung, Stam Verlag Köln 2000, S. 59 ff.

¹⁴⁾ Vgl. Bleßmann, Jörg / Büttner, Artur / Dax, Erwin, Basiswissen IT-Berufe, Anwendungsentwicklung, Stam Verlag Köln 2000, S. 63 ff.

4.5.1. Folge (Sequenz)

Man verwendet eine Sequenz, wenn eine Problemlösung es erfordert, dass mehrere Anweisungen / Verarbeitungen hintereinander auszuführen sind. Beim Flussdiagramm (PAP) werden die Verarbeitungen durch kleine Kästchen und der eigentliche (Programm- / Daten-) Fluss durch Pfeile symbolisiert. Bei Struktogrammen bestehen sie aus ineinander geschachtelten Kästchen. Der Fluss geht bei Struktogrammen immer von der oberen zur unteren Querseite.

Programmablaufplan nach DIN 66 001 – A	Struktogramm nach DIN 66 261 - A	Pseudocode (Beispiel, keine Norm vorhanden)
		<pre>V1 V2</pre>

4.5.2. Auswahl (Selektion)

Bei der Auswahl werden Verarbeitungen in Abhängigkeit von einer oder mehreren Bedingungen ausgeführt. An Stelle der Bezeichnung Auswahl wird oft auch die Bezeichnung Selektion, Alternative oder auch Verzweigung verwendet.

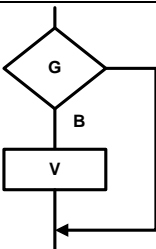
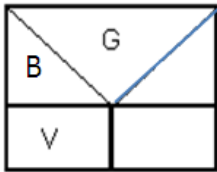
Man unterscheidet dabei die

- bedingte Verarbeitung
- einfache Alternative
- mehrfache Alternative.

Bedingte Verarbeitung

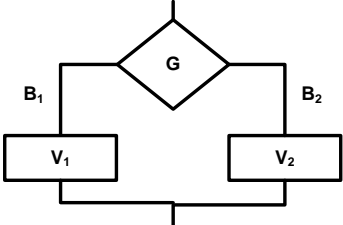
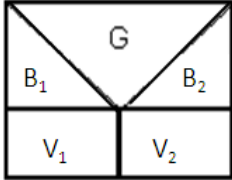
Die bedingte Verarbeitung macht die Ausführung einer Verarbeitung von nur einer Bedingung (Abk. G) abhängig. Ist diese erfüllt, wird die Verarbeitung ausgeführt und das Programm anschließend fortgesetzt. Ist die Bedingung nicht erfüllt, wird die Verarbeitung nicht ausgeführt und das Programm direkt fortgesetzt.

In der Regel wird eine bedingte Verarbeitung mit Hilfe der Schlüsselwörter **Falls und dann** bzw. **IF und then** beschrieben. Softwarewerkzeuge orientieren sich an diesen **Schlüsselwörtern**.

Programmablaufplan nach DIN 66 001 – PA	Struktogramm nach DIN 66 261 - A	Pseudocode (Beispiel, keine Norm vorhanden)
		<pre>if G = B then V end if</pre>

Einfache Alternative

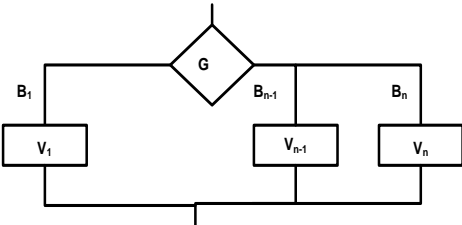
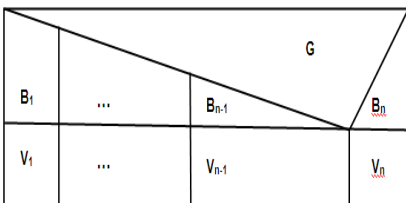
Die einfache Alternative ermöglicht die Auswahl von einer aus zwei Möglichkeiten. Dabei wird die Auswahl von einer gemeinsamen Bedingung (Abk. G) abhängig gemacht.

Programmablaufplan nach DIN 66 001 – PA	Struktogramm nach DIN 66 261 - A	Pseudocode (Beispiel, keine Norm vorhanden)
		<pre>if G = B₁ then V₁ else V₂ end if</pre>

Ist die Bedingung erfüllt, wird die Verarbeitung 1 ausgeführt und anschließend das Programm fortgesetzt. Ist die Bedingung nicht erfüllt, wird die Verarbeitung 2 ausgeführt und anschließend das Programm fortgesetzt.

Die mehrfache Alternative

Die mehrfache Alternative erlaubt die bedingte Auswahl von genau einer aus mehreren Alternativen. Die Auswahl der entsprechenden Verarbeitung wird dabei vom Wert eines Bedingungsausdrucks (dem so genannten Selektor, Abk. G) abhängig gemacht.

Programmablaufplan nach DIN 66 001 – PA	Struktogramm nach DIN 66 261 - A	Pseudocode (Beispiel, keine Norm vorhanden)
		<pre>case G = B1: V₁ B2: V₂ B3: V₃ ... B_{n-1}: V_{n-1} B_n: V_n end case</pre>

In einigen Programmiersprachen (z. B. C++, C#) lässt sich nicht jede Alternative mit mehreren Möglichkeiten als mehrfache Alternative formulieren. Dabei machen Abfragen bezüglich Intervalle Probleme (beispielsweise gewährte Zinssätze für Kapitalanlagen zwischen 1.000€ und 9.000€). Diese wird mittels Kombinationen von Alternativen gelöst. Wir geben hier eine Möglichkeit einer verschachtelten Alternative in Pseudocode-Notation an:

Die verschachtelte Alternative

```
if G = B1 then
    V1
else if G = B2 then
    V2
else if G = B3 then
    V3
else
    V4
end if
```

4.5.3. Wiederholung

Eine Wiederholung macht es möglich, dass bestimmte Programmteile / Verarbeitungen mehrfach durchlaufen bzw. ausgeführt werden können. An Stelle der Bezeichnung Wiederholung wird oft auch die Bezeichnung Schleife, Zyklus oder auch Iteration verwendet. Dabei stehen vier verschiedene Grundformen zur Verfügung.

Wiederholung mit vorhergehender Prüfung (kopfgesteuerte Schleife)

Bei der Wiederholung mit vorhergehender Prüfung (kopfgesteuerte Schleife) wird vor jedem Durchlauf die Bedingung ausgewertet. Ist die Bedingung erfüllt, so wird die Verarbeitung abgearbeitet und anschließend versucht, die Schleife erneut zu durchlaufen (Bedingungsprüfung). Ist die Bedingung nicht erfüllt, so wird die Schleife insgesamt beendet bzw. gar nicht erst durchlaufen.

Programmablaufplan nach DIN 66 001 – PA	Struktogramm nach DIN 66 261 - A	Pseudocode (Beispiel, keine Norm vorhanden)
		<pre>while B do V end while</pre>

Die Verarbeitung wird auch als Schleifenkörper bezeichnet. Sie wird ausgeführt, solange die Bedingung erfüllt ist. Dabei ist es wichtig, dass die Bedingung auch einmal den Wert *falsch* liefert (nicht mehr zutrifft), um die Schleifenbearbeitung zu beenden.

Zählschleife

Die Zählschleife ist eine Sonderform der Wiederholung mit vorhergehender Prüfung (kopfgesteuerte Schleife), wobei als Bedingung eine Zählvariable überprüft wird. Eine Zählschleife sollte immer dann verwendet werden, wenn ein vorher feststehender Wertebereich lückenlos durchlaufen werden soll.

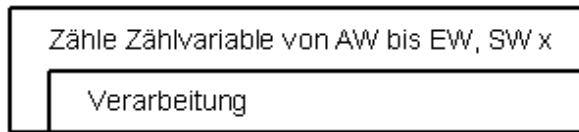
Für Zählschleifen gibt es keine standardisierten Symbole, deshalb geben wir hier ein paar mögliche Darstellungen an:

Pseudocode

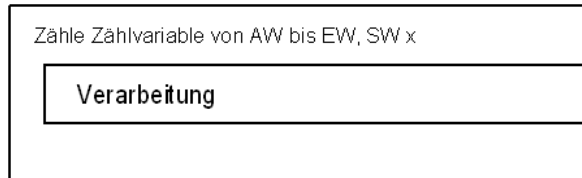
```
for i = AW to EW, step SW do
  V
end for
```

Struktogramm

Erste Möglichkeit (Spezialfall der kopfgesteuerten Schleife):



Zweite Möglichkeit:



Anstelle der Formulierung „Zähle Zählvariable von AW bis EW, SW x“ wird auch die Formulierung „Wiederhole für Zählvariable von AW bis EW (SW x)“ benutzt.

In den Programmiersprachen werden für Zählvariablen oft die Buchstaben i, j, k benutzt.

Begriffserläuterung:

AW: Anfangswert

EW: Endwert

SW: Schrittweite: Die Zählvariable wird meist vom Startwert aus hoch oder herunter gezählt. Die Schrittweite legt fest, um wie viel die Zählvariable nach jedem Schleifendurchlauf nach oben oder nach unten gezählt wird.

Wiederholung mit nachfolgender Prüfung (fußgesteuerte Schleife)

Bei der Wiederholung mit nachfolgender Prüfung (fußgesteuerte Schleife) erfolgt die Auswertung der Bedingung erst nach dem Ende eines Schleifendurchlaufs. Die Verarbeitung im Schleifenkörper wird also auf jeden Fall einmal ausgeführt. Ein Beispiel wäre die Überprüfung einer Tastatureingabe, die in jedem Fall erfolgen muss. War sie fehlerhaft, muss sie erneut eingegeben werden, ansonsten wird die Bearbeitung der Schleife beendet und das Programm fortgeführt.

Programmablaufplan nach DIN 66001 - PA	Struktogramm nach DIN 66261 - A	Pseudocode (Beispiel, keine Norm vorhanden)
		<pre>repeat V until B</pre>

Wiederholung ohne Bedingungsprüfung

Man spricht in diesem Fall von einer Endlosschleife, deren Bearbeitung nur durch einen Eingriff von außen zwangsweise abgebrochen wird. Eine solche Schleife ist sinnvoll für Betriebssysteme, Kommunikations- oder Prozessleitsysteme, die eigentlich unendlich lange ausgeführt werden können und nur durch den Eintritt von Sonderbedingungen beendet werden.

Im **Pseudocode** wird die Wiederholung ohne Bedingungsprüfung durch die Schlüsselworte **Wiederhole** und **für immer** dargestellt.

4.5.4. Überblick Kontrollstrukturen

1. Folge (Sequenz)
2. Auswahl (Selektion, Alternative)
 - bedingte Verarbeitung wenn → dann
 - einfache Alternative wenn → dann → sonst
 - mehrfache Alternative wenn Selektor
 - Wert 1 → dann
 - Wert 2 → dann
 - ...
 - Wert n → dann
 - sonst → dann
3. Wiederholung (Schleife, Iteration)
 - kopfgesteuerte Schleife solange wie ... → wiederhole ...
 - Zählschleife zähle von ... bis ... Schrittweite ...
 - fußgesteuerte Schleife wiederhole ... → solange bis ...

4.5.5. Überblick Notationen für Kontrollstrukturen

Struktogramm	Programmablaufplan (PAP)	Pseudocode
programmiersprachen-unabhängig	programmiersprachen-unabhängig	programmiersprachen-unabhängig
grafisch	grafisch	sprachlich
DIN-Norm	DIN-Norm	keine DIN-Norm

4.5.6. Anwendungsbeispiel Kontrollstrukturen

Anforderungsbeschreibung:

Eine Firma möchte das Gehalt der tausend Mitarbeiter und die gesamten Gehaltskosten berechnen lassen.

Für jeden Mitarbeiter wird seine Gehaltsklasse eingegeben.

Es gilt folgende Gehaltszuweisung:

Gehaltsklasse	Gehalt
A	500
B	2000
C	5000
D	10.000

Für jeden Mitarbeiter soll sein Gehalt ausgegeben werden (z. B.: „Der 34. Mitarbeiter erhält 2000 €“.)

Die insgesamt anfallenden Gehaltskosten sollen berechnet und ausgegeben werden.

(Beispiel: „Die Gehaltskosten betragen insgesamt 500.000 €.“)

Vorüberlegungen:

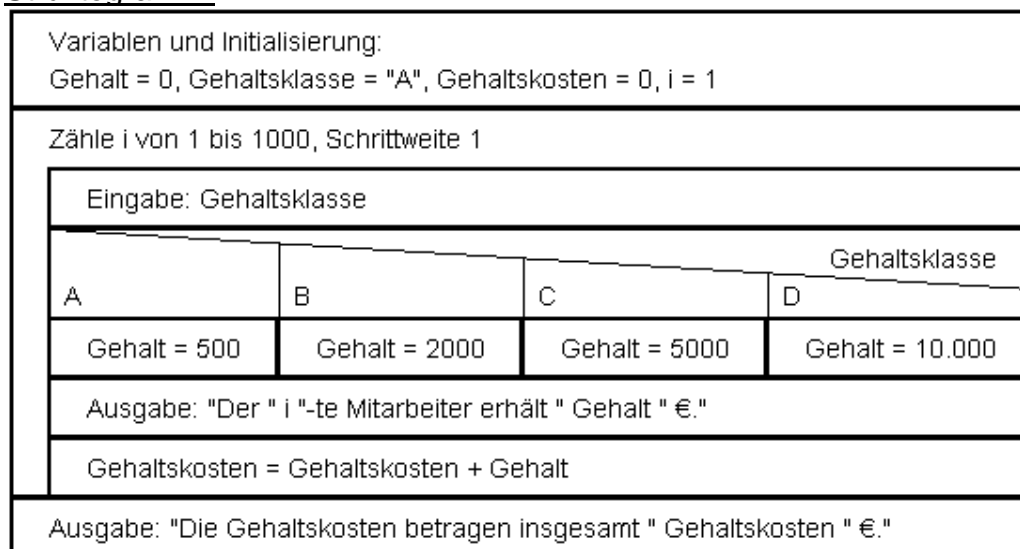
Die Werte von Gehalt, Gehaltskosten und Gehaltsklasse sind variabel.

Gehaltsklasse wird für jeden Mitarbeiter eingegeben und dazu das Gehalt ermittelt.

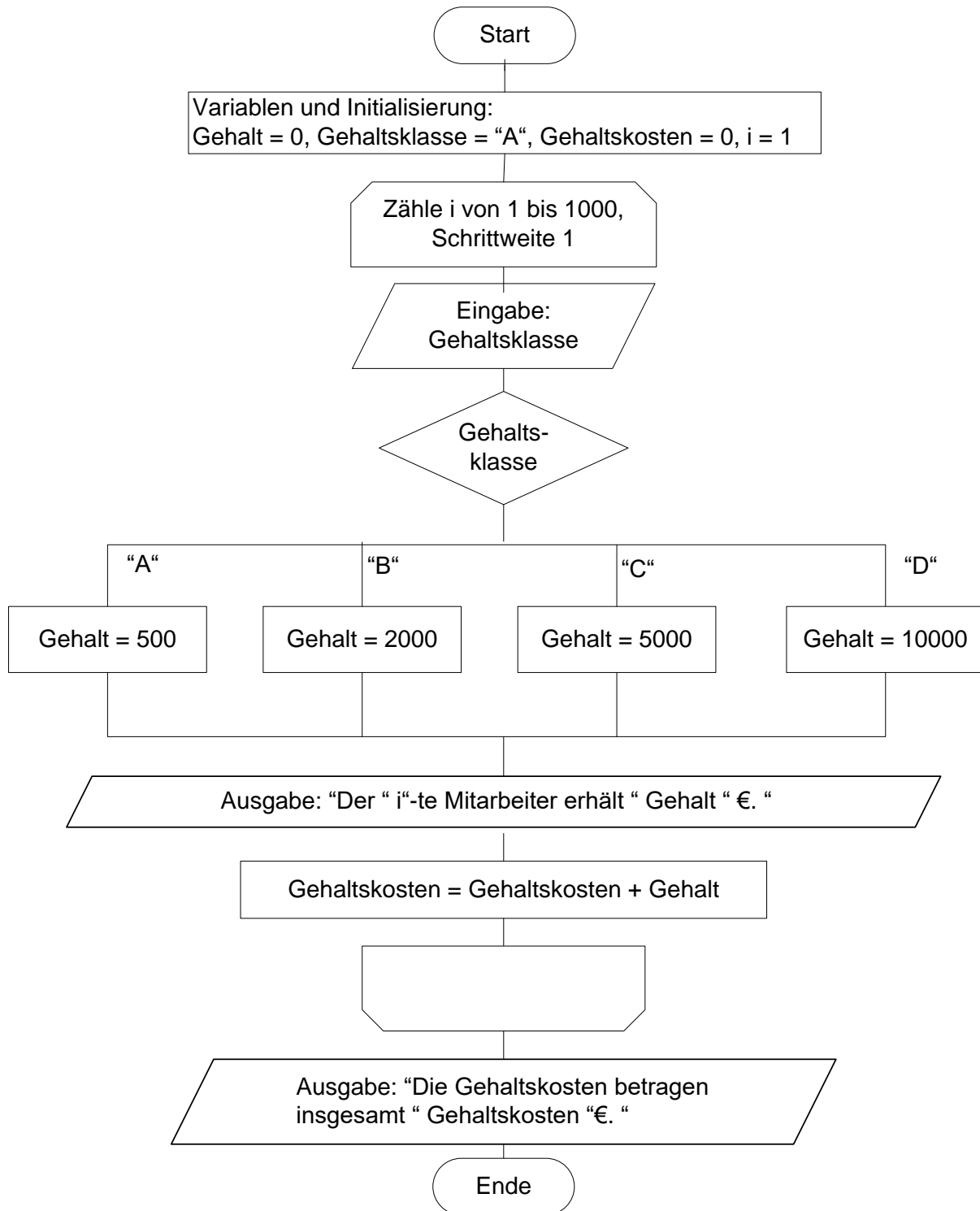
Das aktuelle Gehalt wird zu den bisherigen Gehaltskosten addiert.

Dieses wird wiederholt und daher ist eine Schleife über die Mitarbeiter erforderlich. Da die Anzahl der Durchläufe der Schleife bekannt ist, ist eine Zählschleife die beste Vorgehensweise. Dazu wird noch eine Zählvariable i definiert.

Struktogramm:



Programmablaufplan:



Pseudocode:

In der Pseudocode-Darstellung betrachten wir zusätzlich das Abfangen einer falschen Eingabe in der Variablen „Gehaltsklasse“. In der ersten Variante zeigen wir die Umsetzung einer mehrfachen Alternative, in der zweiten Variante zeigen wir eine verschachtelte Alternative:

Variante 1:

Require: Gehalt = 0.0, Gehaltsklasse = 'A', Gehaltskosten = 0.0, i = 1

for (i = 1 to 1000) **do**

Eingabe Gehaltsklasse

case Gehaltsklasse =

'A': Gehalt = 500.0

'B': Gehalt = 2000.0

'C': Gehalt = 5000.0

'D': Gehalt = 10000.0

default

Gehaltsklasse = 'A'

Gehalt = 500.0

Ausgabe: "Es wurde eine unbekannte Gehaltsklasse eingegeben.

Gehaltsklasse wurde auf 'A' gesetzt."

end case

Ausgabe: "Der" i "-te Mitarbeiter erhält " Gehalt " €."

Gehaltskosten = Gehaltskosten + Gehalt

end for

Ausgabe: "Die Gehaltskosten betragen insgesamt " Gehaltskosten " €."

Variante 2:

Require: Gehalt = 0.0, Gehaltsklasse = 'A', Gehaltskosten = 0.0, i = 1

for (i = 1 to 1000) **do**

Eingabe Gehaltsklasse

if Gehaltsklasse = 'A' **then**

Gehalt = 500.0

else if Gehaltsklasse = 'B' **then**

Gehalt = 2000.0

else if Gehaltsklasse = 'C' **then**

Gehalt = 5000.0

else if Gehaltsklasse = 'D' **then**

Gehalt = 10000.0

else

Gehaltsklasse = 'A'

Gehalt = 500.0

Ausgabe: "Es wurde eine unbekannte Gehaltsklasse eingegeben.

Gehaltsklasse wurde auf 'A' gesetzt."

end if

Ausgabe: "Der" i "-te Mitarbeiter erhält " Gehalt " €."

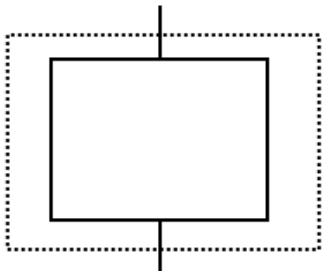
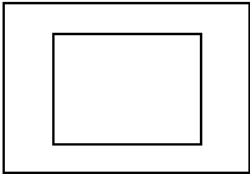

Gehaltskosten = Gehaltskosten + Gehalt

end for

Ausgabe: "Die Gehaltskosten betragen insgesamt " Gehaltskosten " €."

4.5.7. Funktionen als Bausteine (Blockbildung)¹⁵

Große Algorithmen kann man in Teilfunktionen oder auch Blöcke aufsplitten. Das Ziel von Funktionen ist es, durch Modularisierung die Zeit für die Entwicklung von Programmen gering zu halten. Funktionen (Blöcke) kann man sich am besten als Spezialisten vorstellen, die immer dann gerufen werden, wenn eine bestimmte Aufgabe zu erledigen ist. Arbeitet eine Funktion einmal einwandfrei, kann sie immer wieder auch in anderen Programmen verwendet werden. Um den Einsatz von Funktionen auch in anderen Programmen zu gewährleisten, sollte man bestrebt sein, die Funktionen so allgemein gültig wie möglich zu halten! Der Aufruf von Funktionen erfolgt dabei über ihren Namen mit einer Anzahl von Parametern. Dadurch können Funktionen mehr als nur eine festgelegte Aufgabe erfüllen. Man kann Funktionen flexibler gestalten bzw. handhaben, indem man ihnen Werte, so genannte Parameter, übergibt. Diese legen genau fest, was gemacht werden soll, bzw. womit die Funktion arbeiten soll. Offiziell gibt es keine Symbole für die Darstellung von Funktionen (Blöcken). Im praktischen Gebrauch findet man allerdings folgende Darstellungen:

Programmablaufplan nach DIN 66 001 – PA	Struktogramm nach DIN 66 261 - A	Pseudocode (Beispiel, keine Norm vorhanden)
	 Weitere Schreibweise: 	Definition Funktionskopf: Rückgabebetyp Funktionsname (Datentyp Übergabeparameter); Funktionsaufruf: Funktionsname(aktuelle Werte für Übergabeparameter)

¹⁵ Vgl. Erlenkötter, Helmut, *C Programmieren von Anfang an*. Reinbeck bei Hamburg: Rohwohlt Taschenbuch Verlag, 1999, S. 80.

5. Datenflussplan

Ein Datenflussplan (auch data flow chart) ist eine grafische Übersicht, welche die Programme und Daten, die zu einer Gesamtaufgabe gehören, miteinander verbindet. Er zeigt, welche Teile eines Programms von Daten durchlaufen werden und welche Art der Bearbeitung innerhalb der Programme vorgenommen wird.

Gerade in Großunternehmen ist es noch häufig der Fall, dass verschiedene Abteilungen nicht vernetzt sind und nicht gegenseitig auf Daten zugreifen können. Um diese Situationen zu analysieren und ggf. zu verändern eignet sich die Anfertigung von Datenflussplänen.

Der Datenflussplan enthält

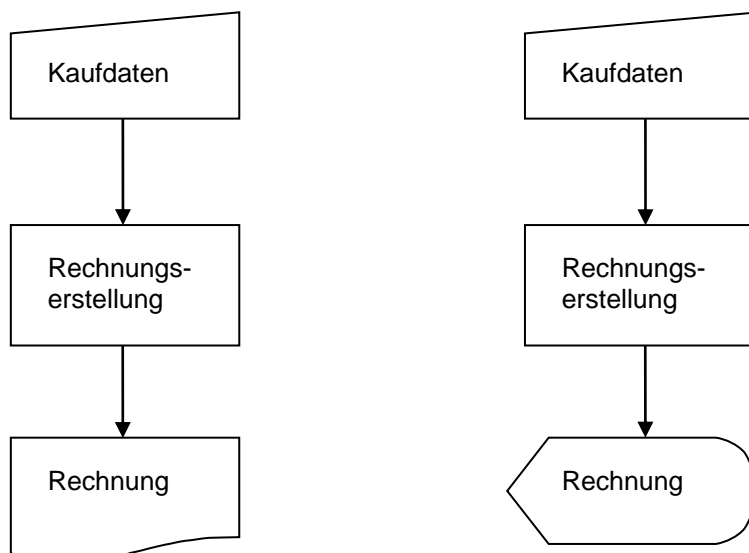
- alle Programme
- alle Datenträger, Dateien, Bänder, Ausdrucke
- manuelle Abläufe
- Form der Verwendung Lesen/Schreiben
- Kommentare, Bemerkungen.

Die Symbole des Datenflussplans nach DIN 66001¹⁶ beschreiben

- Bearbeitungsvorgänge
- Datenträger bei der Ein-/Ausgabe
- Flusslinie der Daten (Linien mit Pfeilspitzen).

Beispiel 1:

Grundlegende Symbole zur Verarbeitung von Daten und Ausgabe auf dem Drucker bzw. auf dem Bildschirm (Rechnung für einen Artikelkauf)

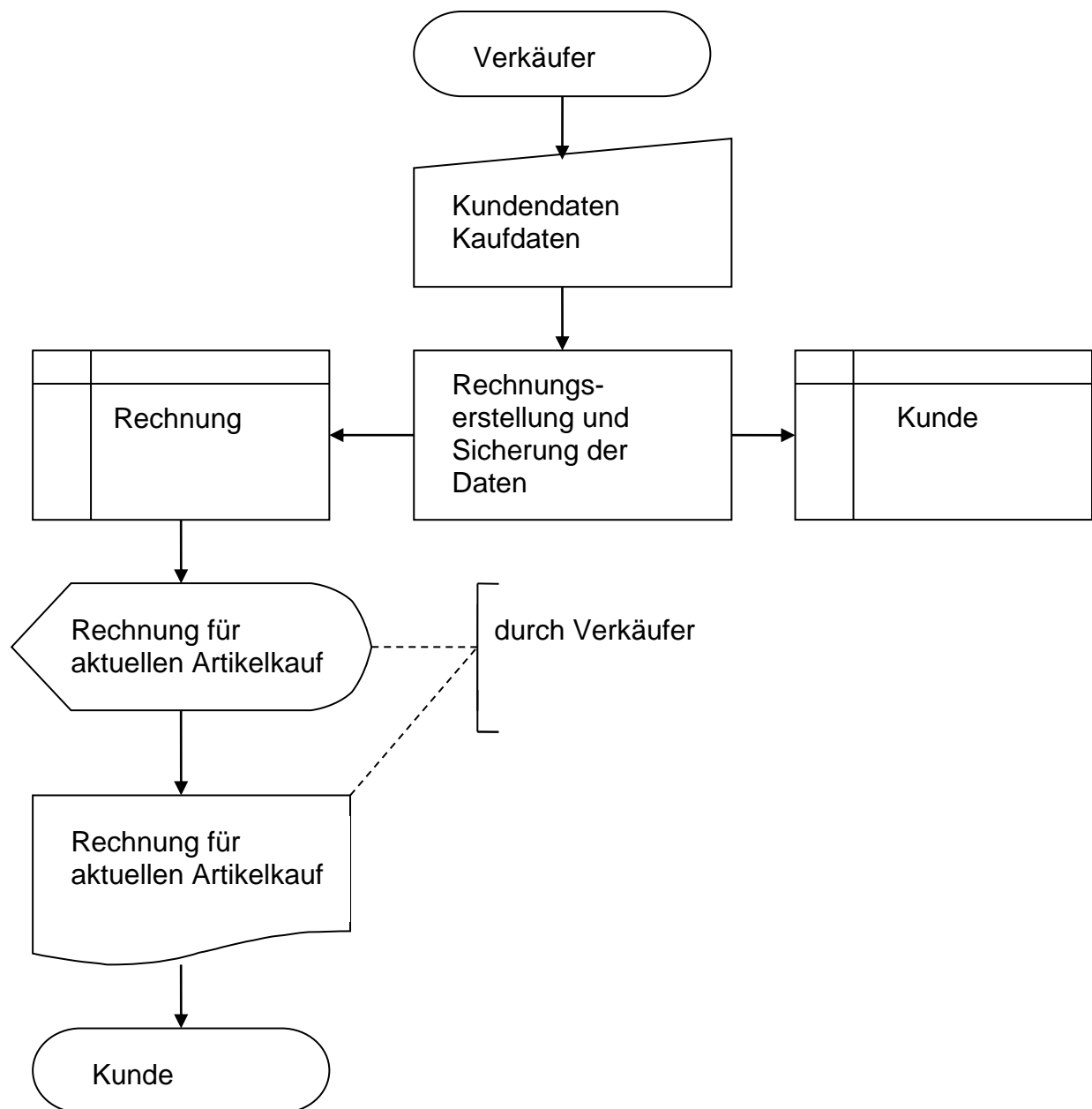


¹⁶ Vgl. Bleßmann, Jörg; Büttner, Artur; Dax, Erwin, Basiswissen IT-Berufe - Anwendungsentwicklung. Köln: Stam Verlag, 2000, S. 57 f.

Beispiel 2:

Ablauf einer Datenerfassung und Rechnungserstellung für einen Artikelkauf

Der Verkäufer gibt an seinem Rechner die relevanten Daten für den Kauf eines Artikels ein. Diese sind Daten zum Kunden (z. B. Kundennummer) und Kaufdaten (z. B. Artikelnummer, Anzahl, Stückpreis). Sobald er auf den Button „Rechnung erstellen“ klickt, wird automatisch ein Verarbeitungsprogramm gestartet. Dieses berechnet den Gesamtpreis, speichert die Rechnungsdaten im Rechnungszentralspeicher bzw. die kundenrelevanten Daten im Kundenzentralspeicher. Der Verkäufer lässt sich die Rechnungsdaten am Bildschirm anzeigen und druckt die Rechnung aus. Den Rechnungsausdruck erhält der Kunde.

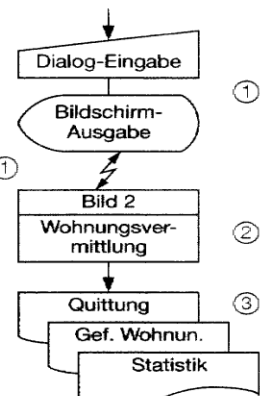


Symbole und ihre Regeln¹⁷

Symbol	Bedeutung	Symbol	Bedeutung	Symbol	Bedeutung
	Verarbeitung, Verarbeitungseinheit		Steuerung der Verarbeitungsfolge von ausen		Daten auf Lochstreifen, Lochstreifeneinheit
	Manuelle Verarbeitung, Verarbeitungsstelle		Daten, allgemein Datenträgereinheit, allgemein		Daten auf Speicher mit auch direktem Zugriff, Datenträgereinheit
	Verzweigung, Auswahlinheit		Maschinell zu verarbeitende Daten, Datenträgereinheit		Daten im Zentralspeicher, Zentralspeicher
	Schleifenbegrenzung		Manuell zu verarbeitende Daten, Manuelle Ablage (z.B. Archiv, Ziehkana)		Manuelle optische oder akustische Eingabedaten, Eingabeeinheit
	Anfang		Daten auf Schriftstück (z.B. auf Belegen, Mikrofilm) Ein-/Ausgabeeinheit		Verbindung, Verarbeitungsfolge, Zugriffsmöglichkeit
	Ende		Daten auf Speicher mit nur sequentiellm Zugriff, Datenträgereinheit		Verbindung zur Datenübertragung, Datenübertragungsweg
	Synchronisierung paralleler Verarbeitungen		Maschinell erzeugte optische oder akustische Daten, Ausgabeeinheit		Grenzstelle (zur Umwelt), z.B. Anfang, Ende
	Sprung mit/ohne Rückkehr		Daten auf Karte (z.B. Lochkarte, Magnetkarte), Lochkarteneinheit		Verbindungsstelle
	Unterbrechung einer anderen Verarbeitung		Hinweis auf ergänzende detaillierte Darstellungen		Verfeinerung
			Hinweis auf weitere Dokumentationsstellen		Bemerkung

Regeln zur Erstellung von Plänen

- Pfeile geben die Flussrichtung an.
- Zwischen Sinnbildern dürfen mehrere Verbindungen verlaufen.
- Kreuzungen von Verbindungslinien vermeiden.
- Hintereinander gezeichnete Sinnbilder gleicher Art bilden eine Einheit mehrerer gleichartiger Datenträger. ③
- Sinnbilder können miteinander verknüpft werden, z. B. zu einer Ausgabeeinheit. ①
- Innenbeschriftungen sollen weitere Abläufe erkennen lassen und eindeutig zuordnen.
- Bezeichnung erfolgt oben links des Sinnbildes.
- Durch einen Querstrich oben im Sinnbild wird auf eine detaillierte Darstellung derselben Dokumentation hingewiesen, z. B. schrittweise Verfeinerung eines Programmablaufs. ②
- Mit zusätzlichen senkrechten Linien in den Sinnbildern „Daten“ und „Verarbeitung“ wird auf eine Dokumentation an anderer Stelle hingewiesen.



¹⁷ Vgl. Hübscher, Heinrich u. a., IT-Handbuch. Braunschweig: Westermann Schulbuch Verlag, 2000, S. 223.