

Wilfried Koch

***Professionelles Programmieren
von Anfang an***

***mit Free Pascal
und der freien Entwicklungsumgebung
Lazarus***

Teil 1

In der Reihe „Informatik ganz einfach“ (Herausgeber Prof. Dr.-Ing. Wilfried Koch, Hochschule Ravensburg-Weingarten) sind bisher folgende Bücher erschienen:

Band 1 C++Builder Rezeptbuch – Teil 1

Band 4 C++Builder Rezeptbuch – Teil 2

In der selben Reihe sind für die nächste Zeit folgende Neuerscheinungen geplant:

Band 2 Multimedialprogrammierung leicht gemacht – Teil 1 Video

Band 5 Multimedialprogrammierung leicht gemacht – Teil 2 Audio

*Band 6 Professionelles Programmieren
von Anfang an - Teil 2*

*Um den Zugangscode für die Serviceseiten dieses Buches zu erhalten
Senden Sie bitte eine Kopie Ihres Kaufbelegs an okomedien@gmx.de.*

Wilfried Koch

Professionelles Programmieren von Anfang an

*mit Free Pascal und der
freien Entwicklungsumgebung Lazarus*

*Ein praxisnahes Lehrbuch für
Ein- und Umsteiger*

Teil 1

*Pascal
Entwicklungsumgebung Lazarus
Grundlagen der
objektorientierten Programmierung
Einfache Grafikanwendungen*

Reihe „Informatik ganz einfach“

Band 3

Obwohl sämtliche in diesem Buch enthaltenen Darstellungen, Verfahren und Informationen nach bestem Wissen zusammengestellt und mit Sorgfalt getestet wurden, sind Fehler nicht vollständig auszuschließen. Deshalb sind die in diesem Buch enthaltenen Informationen mit keiner Garantie oder Verpflichtung verbunden. Autor und Verlag übernehmen deshalb keinerlei Verantwortung und werden deshalb keine daraus folgende oder sonstige Haftung für Schäden übernehmen, die auf beliebige Art aus der Benutzung dieses Buchs oder Teilen davon entsteht.

In gleicher Weise übernehmen Verlag und Autor keine Gewähr dafür, dass die beschriebenen Verfahren frei von Schutzrechten Dritter sind. Die Wiedergabe von Warenbezeichnungen, Gebrauchsnamen, Markennamen usw. in diesem Buch berechtigt deshalb auch ohne besondere Kennzeichnung nicht dazu, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann in beliebiger Weise benutzt werden dürften.

Die Deutsche Bibliothek – CIP Einheitsaufnahme

Ein Titelsatz für diese Publikation ist bei der Deutschen Bibliothek erhältlich.

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdrucks und der Vervielfältigung des Buches oder Teilen daraus vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder auch ein anderes Verfahren) – auch nicht für Zwecke der Unterrichtsgestaltung – reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2016 Wilfried Koch

Herstellung: Books on Demand GmbH, Norderstedt

Verlag: Oberkochener Medienverlag, Oberkochen

Printed in Germany

ISBN 978-3-945899-01-4

Die URLs zum Buch:

www.informatik-ganz-einfach.de

www.okomedien.de/buecher/informatik-ganz-einfach/laz1

Inhaltsverzeichnis (Schnellübersicht)

1. Hintergrund und Anwendung dieses Buches.....	17
2. „Hallo Welt“ - Ihr erstes Programm.....	23
3. Von Typen, Variablen, Bezeichnern, Operatoren und ähnlichem – der Weg zu Ihrem ersten wirklich nützlichen Programm.....	50
4. So kommt die Logik in Ihr Programm – Logisches, Boolesches und Relationales.....	85
5. Damit Ihre Arbeit einfacher wird: Standardprozeduren und -funktionen.....	96
6. Es geht nicht immer nur geradeaus - Steuerstrukturen.....	116
7. Ihr Programm braucht Umgebungskontakt - Daten-Ein- und Ausgabe, Speichern von Texten.....	162
8. Programmieren mit Bausteinen – Prozeduren, Funktionen und Units.....	181
9. Die Bedienoberfläche „verkauft“ Ihr Programm – erste Schritte in Richtung grafische Bedienoberflächen.....	221
10. Die liebe Verwandtschaft – Datenfelder (arrays).....	253
11. Noch mehr Typen – Zähltypen, Teilbereichstypen und Mengen.....	275
12. Record-Verdächtiges – Arbeiten mit komplexen Datenstrukturen.....	290
13. Sie wissen nicht was noch kommt – Dynamische Daten.....	303
14. Am Ende ist nicht alles vorbei – Einfache Persistenzlösungen.....	330
15. Die Welt besteht aus Klassen und Objekten – Grundlagen der objektorientierten Programmierung.....	342
16. Ein Bild sagt mehr als 1000 Worte – Einfache Grafikprogrammierung.....	361
17. Geben Sie es der Grafik zurück – Interaktive Grafikprogrammierung.....	401
18. Denn was man schwarz auf weiß besitzt.... – Arbeiten mit dem Drucker.....	425

Inhaltsverzeichnis

1. Hintergrund und Anwendung dieses Buches.....	17
1.1. Zielsetzung.....	17
1.2. Programmieren – worum geht es dabei überhaupt?.....	17
1.2.1. Programmiersprachen.....	17
1.2.2. Syntax und Semantik.....	18
1.2.3. Vom Compiler zur Integrierten Entwicklungsumgebung.....	19
1.3. Pascal – Ein kurzer historischer Überblick.....	20
1.3.1. Der Anfang.....	20
1.3.2. Wichtige Implementationen.....	20
1.3.3. Free Pascal und Lazarus.....	20
1.4. Didaktische Ausrichtung.....	21
1.5. Wenn Sie nicht Seite für Seite vorwärts arbeiten wollen.....	22
2. „Hallo Welt“ - Ihr erstes Programm.....	23
2.1. Aufgabe: Erstellen Sie eine (Konsol-) Anwendung.....	23
2.2. Der Beitrag von Pascal.....	23
2.2.1. Aufbau eines Programms.....	23
2.2.2. Syntaxdiagramme (Fahrprogramme).....	25
2.2.3. Darstellung von Texten.....	27
2.2.4. Ausgabeanweisung.....	27
2.2.5. Elementare Eingabeanweisung.....	28
2.3. So nützen Sie Lazarus.....	29
2.3.1. Ohne Installation geht gar nichts.....	30
2.3.2. Start der IEU von Lazarus.....	39
2.3.3. Die Bedienoberfläche (Standardeinstellung).....	39
2.3.4. Auswahl einer Programmschablone.....	41
2.3.5. Ergänzung des Programmrahmens mit dem Quelltexteditor.....	42
2.3.6. Speichern und Laden von Dateien und Projekten.....	45
2.3.7. Übersetzen Sie Ihr Projekt.....	46
2.4. Lösung.....	47
2.4.1. So erstellen Sie Ihr Programm.....	47
2.4.2. Kommentare.....	48
2.5. Übungsaufgaben zu Kapitel 2.....	49
2.5.1. Änderung des Ausgabetextes.....	49

2.5.2. Langer Ausgabertext.....	49
2.6. Lösungen zu den Übungsaufgaben zu Kapitel 2.....	49
2.6.1. Lösung Änderung des Ausgabetextes.....	49
2.6.2. Lösung Langer Ausgabertext.....	49
3. Von Typen, Variablen, Bezeichnern, Operatoren und ähnlichem – der Weg zu Ihrem ersten wirklich nützlichen Programm.....	50
3.1. Aufgabe: Elementares Rechnen mit ganzen Zahlen.....	50
3.1.1. Aufgabenstellung: Ermittlung von Polynomwerten.....	50
3.1.2. Der Beitrag von Pascal.....	51
3.1.3. Lösung.....	65
3.2. Aufgabe: Elementares Rechnen mit Gleitkommazahlen.....	66
3.2.1. Aufgabenstellung: Ermittlung von Polynomwerten.....	66
3.2.2. Der Beitrag von Pascal.....	66
3.2.3. Lösung.....	74
3.2.4. Weitere Aufgaben.....	75
3.3. Aufgabe: Elementare Textverarbeitung.....	80
3.3.1. Aufgabenstellung.....	80
3.3.2. Der Beitrag von Pascal.....	80
3.3.3. Der Beitrag von Lazarus.....	81
3.3.4. Lösung.....	83
4. So kommt die Logik in Ihr Programm – Logisches, Boolesches und Relationales.....	85
4.1. Aufgabenstellung.....	85
4.1.1. Schloss und Schlüssel (Umgang mit Bitmustern).....	85
4.1.2. Boolesche Algebra.....	85
4.1.3. Vergleiche von ganzen Zahlen, Gleitkommazahlen und Texten.....	85
4.2. Der Beitrag von Pascal.....	86
4.2.1. Logische Operatoren und Ausdrücke.....	86
4.2.2. Der Datentyp boolean, Boolesche Operatoren und Ausdrücke.....	88
4.2.3. Relationale Operatoren und Ausdrücke.....	89
4.3. Lösungen.....	90
4.3.1. Schloss und Schlüssel.....	90
4.3.2. DeMorgansche Regeln.....	91
4.3.3. Hilfe mein Computer rechnet verkehrt – Relationale Ausdrücke, Informationsdarstellung und andere Feinheiten.....	93
5. Damit Ihre Arbeit einfacher wird: Standardprozeduren und -funktionen.....	96

5.1. Aufgabenstellung.....	96
5.1.1. Verschiedenes aus Mathematik und Physik.....	96
5.1.2. Ermittlung der numerischen Codierung von Textzeichen.....	97
5.1.3. Handhabung von Zeichenketten.....	97
5.1.4. Verzeichnis- und Dateibehandlung.....	98
5.1.5. Datums- und Zeitbehandlung.....	98
5.2. Der Beitrag von Pascal.....	98
5.2.1. Units der Laufzeitbibliothek (Paket RTL).....	99
5.2.2. Mathematik.....	102
5.2.3. Behandlung von Textketten.....	106
5.2.4. Dateibehandlung.....	107
5.2.5. Zeit- und Datumsangaben.....	108
5.2.6. Automatische Typenkonvertierung.....	109
5.3. Lösungen.....	109
5.3.1. Verschiedenes aus Mathematik und Physik.....	109
5.3.2. Ermittlung der numerischen Codierung von Textzeichen.....	111
5.3.3. Handhabung von Zeichenketten.....	111
5.3.4. Aufgaben zur Verzeichnis- und Dateibehandlung.....	113
5.3.5. Datums- und Zeitbehandlung.....	114
6. Es geht nicht immer nur geradeaus - Steuerstrukturen.....	116
6.1. Einfache Fallunterscheidung.....	117
6.1.1. Aufgabe: Werte einer abschnittsweise definierten Funktion berechnen.....	117
6.1.2. Beitrag von Pascal: Bedingte Anweisungen (Verzweigungen).....	118
6.1.3. Der Beitrag von Lazarus - Syntaxsensitiver Editor.....	123
6.1.4. Der Beitrag von Lazarus – Debugger zum Ersten.....	125
6.1.5. Lösung.....	130
6.1.6. Weitere Aufgaben.....	131
6.2. Mehrfachverzweigungen.....	133
6.2.1. Aufgabe: Gewinnhöhe in Abhängigkeit von der Losnummer ermitteln.....	133
6.2.2. Aufgabe: Erstellung eines Rechentrainers.....	134
6.2.3. Der Beitrag von Pascal.....	134
6.2.4. Der Beitrag von Lazarus – Debuggen von Mehrfachverzweigungen.....	136
6.2.5. Lösungen.....	138
6.3. Wiederholte Ausführung von Anweisungen.....	143
6.3.1. Aufgaben zur wiederholten Ausführung von Anweisungen.....	143
6.3.2. Der Beitrag von Pascal - Wiederholte Ausführung von Operationen	

(Schleifenbildung).....	144
6.3.3. Der Beitrag von Lazarus – Eingabehilfen des Editors.....	152
6.3.4. Der Beitrag von Lazarus – Debuggen von Schleifen.....	153
6.3.5. Lösungen.....	154
6.4. Sonstige Möglichkeiten der Programmsteuerung.....	157
6.4.1. Aufgaben, die besondere Steuerkonstrukte einsetzen.....	157
6.4.2. Der Beitrag von Pascal - Sonstige Möglichkeiten der Programmsteuerung	158
6.4.3. Lösung.....	159
7. Ihr Programm braucht Umgebungskontakt - Daten-Ein- und Ausgabe, Speichern von Texten.....	162
7.1. Aufgabenstellung.....	162
7.1.1. Erstellen, anzeigen und abspeichern einer Textinformation.....	162
7.1.2. Einlesen und Anzeigen auf der Festplatte gespeicherter Texte.....	163
7.2. Der Beitrag von Pascal (klassisch prozedural).....	163
7.2.1. Schreib- und Leseanweisungen.....	163
7.2.2. Dateiartern.....	165
7.2.3. Textdateien.....	165
7.2.4. Prozeduren und Funktionen für die Dateibearbeitung.....	165
7.3. Der Beitrag von Lazarus.....	171
7.3.1. Fehlerbehandlung.....	171
7.3.2. Abschlussbehandlung.....	177
7.3.3. Besondere Aspekte des Debuggens.....	177
7.4. Lösung.....	178
7.4.1. Erstellen, anzeigen und abspeichern einer Textinformation.....	178
7.4.2. Einlesen und Anzeigen auf der Festplatte gespeicherter Texte.....	179
8. Programmieren mit Bausteinen – Prozeduren, Funktionen und Units.....	181
8.1. Prozeduren (Benutzerdefinierte Anweisungen).....	181
8.1.1. Aufgaben.....	181
8.1.2. Der Beitrag von Pascal.....	181
8.1.3. Der Beitrag von Lazarus.....	185
8.1.4. Lösung.....	186
8.2. Funktionen.....	189
8.2.1. Aufgaben.....	189
8.2.2. Der Beitrag von Pascal.....	189

8.2.3. Der Beitrag von Lazarus.....	191
8.2.4. Lösung.....	192
8.3. Implementationspraxis von Funktionen und Prozeduren.....	193
8.4. Verschachtelte Unterprogramme.....	195
8.4.1. Gültigkeit von Bezeichnern bei verschachtelten Unterprogrammen.....	195
8.4.2. Aufruf geschachtelter Unterprogramme.....	197
8.4.3. Die forward-Deklaration.....	198
8.5. Rekursion - Prozeduren und Funktionen arbeiten für sich selbst.....	198
8.5.1. Aufgaben.....	198
8.5.2. Der Beitrag von Pascal.....	199
8.5.3. Lösung.....	200
8.5.4. Der Beitrag von Lazarus.....	201
8.5.5. Lösung.....	203
8.6. Funktions- und Prozedurvariable.....	203
8.6.1. Aufgabe.....	203
8.6.2. Der Beitrag von Pascal.....	204
8.6.3. Der Beitrag von Lazarus.....	207
8.6.4. Lösung.....	208
8.7. Units.....	208
8.7.1. Aufgaben.....	209
8.7.2. Der Beitrag von Pascal.....	209
8.7.3. Der Beitrag von Lazarus.....	214
8.7.4. Lösung.....	218
9. Die Bedienoberfläche „verkauft“ Ihr Programm – erste Schritte in Richtung grafische Bedienoberflächen.....	221
9.1. Objektorientierte Programmierung ganz kurz.....	221
9.2. Umstellung der Programme aus den Kapiteln 2 und 3 auf eine grafische Bedienoberfläche.....	223
9.3. Der Beitrag von Pascal.....	223
9.4. Der Beitrag von Lazarus.....	223
9.4.1. Anwendung mit grafischer Bedienoberfläche.....	225
9.4.2. Objektinspektor.....	228
9.4.3. Komponenten für die Datenausgabe.....	235
9.4.4. Komponenten für die Programmsteuerung.....	237
9.4.5. Komponenten für die Dateneingabe.....	240

9.4.6. Komponenten für die Textbearbeitung.....	247
9.4.7. Komponenten zur Gliederung der Fenster.....	249
9.5. Nicht alles, was Sie für Ihre Arbeit benötigen, kann in diesem Buch stehen – Die Hilfefunktionen.....	251
10. Die liebe Verwandtschaft – Datenfelder (arrays).....	253
10.1. Aufgabenstellung.....	253
10.1.1. Arbeiten mit einer Messwerttabelle.....	253
10.1.2. Gliederung von Umsätzen nach Umsatzart und -monat.....	254
10.1.3. Lösung eines Linearen Gleichungssystems.....	255
10.2. Der Beitrag von Pascal.....	255
10.2.1. Strukturierte Datentypen.....	255
10.3. Datenfelder.....	256
10.3.1. Feldgrenzen.....	257
10.3.2. Feldtypen.....	260
10.4. Der Beitrag von Lazarus.....	260
10.4.1. Bedienoberfläche.....	260
10.5. Lösungen.....	262
10.5.1. Aufgabe Messwerttabelle.....	262
10.5.2. Aufgabe Umsatzübersicht.....	264
10.5.3. Aufgabe Lösung eines Linearen Gleichungssystems.....	266
11. Noch mehr Typen – Zähltypen, Teilbereichstypen und Mengen.....	275
11.1. Aufgabenstellung.....	275
11.1.1. Selbstdokumentierender Quelltext, Verbesserung des Dokumentationswerts	275
11.1.2. Betrachtung von Unterbereichen.....	275
11.1.3. Kombination von Eigenschaften.....	276
11.2. Der Beitrag von Pascal.....	276
11.2.1. Aufzählungstypen (enumerated Types).....	276
11.2.2. Teilbereichstypen (subrange types).....	279
11.2.3. Mengentypen.....	281
11.2.4. Die for - in - Schleife.....	285
11.3. Der Beitrag von Lazarus.....	286
11.4. Lösung.....	287
11.4.1. Aufzählungstypen.....	287
11.4.2. Verbesserung des Dokumentationswerts durch Nutzung von Teilbereichs-	

und Aufzählungstypen.....	287
11.4.3. Teilbereichstypen.....	288
11.4.4. Mengen.....	288
12. Record-Verdächtiges – Arbeiten mit komplexen Datenstrukturen.....	290
12.1. Die Aufgabe.....	290
12.1.1. Komplexe, logisch miteinander verbundene Daten aus der technischen Datenverarbeitung.....	290
12.1.2. Komplexe, logisch miteinander verbundene Daten aus der kommerziellen Datenverarbeitung.....	290
12.2. Der Beitrag von Pascal – Record-Datentypen.....	291
12.2.1. Syntax und Anwendungsmöglichkeiten.....	291
12.2.2. Zuweisungsoperationen auf Records und Teile von Records.....	293
12.2.3. Die with- (Pseudo-) Anweisung.....	295
12.2.4. Record-Konstante.....	296
12.2.5. Varianten-Records.....	297
12.3. Lösung.....	298
12.3.1. Record-Datentyp aus der technischen Datenverarbeitung.....	298
12.3.2. Beispiel für den Record-Datentyp aus der kommerziellen Datenverarbeitung.....	300
12.4. Problemgerechte Datendarstellung durch Verwendung von Records.....	300
12.4.1. Verwendung von Record-Datentypen zur Darstellung von Matrizen und Vektoren.....	300
12.4.2. Ausblick auf erweiterte Möglichkeiten.....	302
12.5. Debugging.....	302
13. Sie wissen nicht was noch kommt – Dynamische Daten.....	303
13.1. Aufgaben.....	303
13.1.1. Wechselnder Speicherplatzbedarf.....	303
13.1.2. Kellerspeicher, Stack (LIFO).....	303
13.1.3. Schieberegister, Warteschlange (FIFO).....	304
13.1.4. Ringspeicher.....	304
13.2. Der Beitrag von Pascal.....	305
13.2.1. Zeiger.....	305
13.3. Lösungen.....	307
13.3.1. Wechselnder Speicherbedarf im Programm.....	307
13.3.2. Lineare Listen.....	311

13.3.3. Kellerspeicher mit Visualisierung.....	313
13.3.4. Warteschlange mit Visualisierung.....	317
13.3.5. Ringspeicher mit Visualisierung.....	321
14. Am Ende ist nicht alles vorbei – Einfache Persistenzlösungen.....	330
14.1. Aufgabe: Lesen und Speichern mit dem Dateisystem.....	330
14.2. Der Beitrag von Pascal.....	330
14.2.1. Definition von Binärdateien.....	331
14.2.2. Verknüpfung der logischen mit der physikalischen Datei.....	331
14.2.3. Öffnen und Schließen.....	331
14.2.4. Schreiben und Lesen.....	331
14.2.5. Positionieren.....	332
14.3. Lösung.....	332
14.3.1. Dateneingabe im Dialog.....	332
14.3.2. Datenspeicherung.....	334
14.3.3. Schreiben auf die Datei.....	334
14.3.4. Lesen von der Datei und Datenanzeige.....	334
14.3.5. Programmcode.....	335
15. Die Welt besteht aus Klassen und Objekten – Grundlagen der objektorientierten Programmierung.....	342
15.1. Aufgabe.....	342
15.2. Der Beitrag von Pascal.....	342
15.2.1. Eigenschaften.....	344
15.2.2. Methoden.....	344
15.2.3. Sichtbarkeit.....	346
15.3. Der Beitrag von Lazarus.....	347
15.3.1. CodeExplorer.....	347
15.3.2. Code-Browser.....	349
15.4. Lösung.....	349
15.4.1. Hauptprogramm ProHaushaltsBuch.....	351
15.4.2. Unit UHaushaltsbuch.....	351
15.4.3. Unit UFrmMainHaushalt.....	355
16. Ein Bild sagt mehr als 1000 Worte – Einfache Grafikprogrammierung.....	361
16.1. Aufgaben.....	361
16.1.1. Darstellung eines Funktionsgraphen.....	361
16.1.2. Turm von Hanoi.....	361

16.2. Der Beitrag von Pascal.....	362
16.3. Der Beitrag von Lazarus.....	362
16.3.1. Zeichenfläche TCanvas.....	362
16.3.2. Schreib- und Zeichenwerkzeuge.....	364
16.3.3. Methoden von TCanvas.....	367
16.3.4. Darstellung von Bitmaps.....	369
16.4. Lösungen.....	372
16.4.1. Darstellung eines Funktionsgraphen.....	372
16.4.2. Türme von Hanoi.....	387
17. Geben Sie es der Grafik zurück – Interaktive Grafikprogrammierung.....	401
17.1. Aufgaben.....	401
17.1.1. Interaktives Zeichnen von Funktionsgraphen.....	401
17.1.2. Grafische Lösung transzendenter Gleichungen.....	401
17.2. Der Beitrag von Pascal.....	402
17.3. Der Beitrag von Lazarus.....	402
17.3.1. Mausereignisse.....	402
17.4. Lösungen.....	402
17.4.1. Interaktives Zeichnen von Funktionsgraphen.....	402
17.4.2. Grafische Lösung transzendenter Gleichungen.....	417
18. Denn was man schwarz auf weiß besitzt... – Arbeiten mit dem Drucker.....	425
18.1. Aufgaben.....	425
18.1.1. Erster Umgang mit dem Drucker.....	425
18.1.2. Drucken eines einfachen Texts.....	425
18.1.3. Drucken eines Texts mit eingebauter Grafik.....	425
18.2. Der Beitrag von Lazarus.....	426
18.2.1. Drucker-Unit Printers.....	426
18.2.2. Dialoge zum Einrichten des Drucks und des Druckers.....	428
18.2.3. Umgang mit JPEG-Dateien.....	430
18.3. Lösungen.....	431
18.3.1. Erster Umgang mit dem Drucker.....	431
18.3.2. Drucker- und Seiteneinstellungen vornehmen und einen Text drucken....	433
18.3.3. Drucken eines Texts mit eingebauter Grafik.....	438

Vorwort

„Schon wieder ein neues Programmierbuch“ mag der eine oder andere stöhnen, der in Werbung oder Buchhandel auf dieses Buch stößt. Meine Antwort darauf ist „ja und zwar aus gutem Grund“, denn es ist durchaus lohnend, sich mit (Free-) Pascal und Lazarus näher zu befassen!

Was spricht für das Arbeiten mit (Free-) Pascal und Lazarus?

Für den Einsteiger bietet Lazarus eine komfortable Entwicklungsumgebung für die Erstellung von Programmen mit grafischer Bedienoberfläche. Lazarus basiert auf der Programmiersprache Pascal bzw. deren Implementation Free Pascal. Pascal wurde nicht zuletzt für Ausbildungszwecke entwickelt und hat aufgrund seiner sauberen Definition seine Eignung für die für die Schul- und Hochschulausbildung unter Beweis gestellt.

Für Umsteiger von Delphi und anderen Programmierungsumgebungen ist Lazarus vor allem dann interessant, wenn man bei der Gestaltung seiner Applikation bisweilen gewisse Kompromisse eingehen kann. Dass Lazarus Delphi momentan, was den Umfang angeht, noch nicht ebenbürtig ist, kann man nicht wegdiskutieren. Man kann jedoch annehmen, dass die stetig wachsende Gemeinschaft der Lazarus-Nutzer und vor allem auch der aktiven Kontributoren daran arbeiten wird, diese Lücke zu schließen.

Lazarus wächst rasch und kontinuierlich

Gerade aufgrund der nach Ansicht des Verfassers derzeit sehr restriktiven Lizenzpolitik von Embarcadero – dem derzeitigen Anbieter von Delphi – bietet die kostenlose Verfügbarkeit von Lazarus nicht nur Schulen und interessierten Privatleute sondern auch Unternehmen die Möglichkeit, Kosten im Bereich der Lizenzierung zu reduzieren und gleichzeitig in einer weitgehend Delphi-kompatiblen Programmierungsumgebung zu arbeiten. Lazarus ist als Multiplattformumgebung angelegt, d. h. dass damit Programme für verschiedene Betriebssystemumgebungen entwickelt werden können. Der Schwerpunkt dieses Buches liegt allerdings beim Einsatz unter Windows.

Lazarus ist eine Multiplattformumgebung. Der Buchschwerpunkt liegt auf aber Windows-Anwendungen

Dieses Buch wendet sich sowohl an Ein- als an Umsteiger. Dabei folgt es einem klaren didaktischen Konzept. Anhand typischer Beispielaufgaben werden Schritt für Schritt Lernziele festgelegt um das Know-How weiter zu entwickeln. Innerhalb einer jeden Aufgabe wird dargelegt, welche Eigenschaften von Pascal und Lazarus für die Bearbeitung der Aufgabe von Bedeutung sind und wie man sie einsetzen sollte. Schließlich wird die komplette Lösung der jeweiligen Aufgabe vorgestellt. Abgeschlossen wird jeder Lernschritt durch Übungsaufgaben, mittels derer das im jeweiligen Kapitel erworbene Wissen vertieft werden kann.

Das klare didaktische Konzept unterstützt **Ein-** und **Umsteiger**.

Dabei werden u. a. folgende Themen behandelt:

- Programmierung in Pascal von den Anfängen bis zu den fortgeschrittenen Feinheiten (Zeiger, Rekursion, Dateiverkehr ...)
- Implementation zahlreicher grundlegender Algorithmen.

- Programmierung von Aufgaben aus dem Mathematikunterricht der Realschulen, der Gymnasien und der Anfangssemester der Hochschulen.
- Pascalentwicklung mit Verwendung einer grafischen Benutzeroberfläche
- Elementare Grafikprogrammierung

Nach dem Studium dieses Buches sind Sie in der Lage ein repräsentatives Spektrum von Aufgaben zu lösen, die durch die Sprachdefinition von (Free-) Pascal abgedeckt sind. Anders als wenn Sie sich ausschließlich auf Free-Pascal stützen würden, werden Ihre Programme fast von Anfang an mit einer professionellen grafischen Bedienoberfläche versehen sein, die sich auf die Softwarebibliothek LCL (Lazarus Component Library) stützt. LCL ist auf Sprachebene (Pascal) mit der von Delphi verwendeten VCL (Visual Component Library) weitgehend schnittstellenkompatibel, sodass Delphi- und Lazarus-Know-How in den meisten Fällen austauschbar wird. Der zweite Teil (Band 6 dieser Reihe „Informatik ganz einfach“) wird weiteren Themen wie Datenbanken, Internet und speziellen Fragen der Systemprogrammierung gewidmet sein.

Frühzeitig wird in diesem Buch auf die Erstellung zeitgemäßer Bedienoberflächen Wert gelegt. Die für die Erstellung der grafischen Bedienoberfläche erforderlichen Kenntnisse in objektorientierter Programmierung werden anfänglich pragmatisch entsprechend dem aktuellen Bedarf vermittelt (Kapitel 9). Später erfolgt dann eine systematische Einführung (Kapitel 15).

Aktuelle Informationen zu diesem Buch, wie Korrekturen, Antworten auf Leserfragen oder auch Bestellmöglichkeiten für CDs mit dem Programmcode zu meinen Büchern finden Sie im Internet unter **www.okomedien.de**

Ein Buch wie dieses kann nicht ohne die tatkräftige Unterstützung Dritter zustande kommen. Ich danke meinen Studenten für die zahlreichen Diskussionen und Anregungen, meinem Sohn Matthias für das Korrekturlesen und die Erprobung der Programmbeispiele und nicht zuletzt meiner Frau Ruth für ihre Geduld.

Aktuelle Informationen zum Thema des Buches erhalten Sie unter der Service-Adresse **www.okomedien.de/service/laz1**. Ihre Zugangsdaten erhalten Sie gegen Übersendung (Email) des Kaufbelegs und Nennung Ihrer Email-Adresse.

Sie brennen sicher schon darauf, Ihre Arbeit mit Lazarus zu beginnen. Laden Sie die Integrierte Entwicklungsumgebung Lazarus aus dem Internet herunter. Installieren und starten Sie Lazarus (näheres finden Sie in 2.3.1) und arbeiten Sie sich dann Schritt für Schritt anhand des Lehrtexts und der Beispiele vorwärts.

Ich wünsche Ihnen eine erfolgreiche Einarbeitung in Lazarus und anschließend eine erfolgreiche Projektarbeit!

Oberkochen im Juni 2016

Wilfried Koch

Die grafische
Bedienoberfläche
gehört
selbstver-
ständlich dazu.

1. Hintergrund und Anwendung dieses Buches

1.1. Zielsetzung

Dieses Buch führt anhand leicht verständlicher Beispiele aus der Schul- und Hochschulausbildung in die Programmierung mit einer auf Pascal basierenden Sprache ein. Es wendet sich einerseits an komplette Programmieranfänger, bietet aber auch Fortgeschrittenen und Umsteigern von anderen Entwicklungsumgebungen die erforderlichen Informationen.

1.2. Programmieren – worum geht es dabei überhaupt?

Ein Programm ist eine Arbeitsvorschrift für einen Rechner. Es gibt – wenigstens im Falle von FreePascal-/Lazarus-Programmen an, welche Arbeitsschritte ein Rechner zur Bearbeitung einer Aufgabe ausführen muss.

Die Erstellung eines Programms ist letztlich die Erstellung einer Arbeitsvorschrift. Arbeitsvorschriften sind Ihnen zur Genüge bekannt: Kochrezepte, Gebrauchsanweisungen, Bastelanweisungen aber auch Musiknoten gehören beispielsweise dazu. Während in den hier genannten Fällen Menschen die Arbeitsvorschriften ausführen, sind es im Falle von Programmen Rechner.

1.2.1. Programmiersprachen

Menschen können nach entsprechender Ausbildung Arbeitsvorschriften, die ihnen in natürlichen Sprachen wie Deutsch, Englisch oder auch Chinesisch aber auch in grafischer Form (z. B. Musiknoten) erteilt werden, korrekt interpretieren. Hingegen „verstehen“ die meisten Rechner bis heute nur Programmiersprachen.

Wenn man Programmiersprachen betrachtet so muss man zwischen den maschinen-nahen (niederen) und den höheren Programmiersprachen unterscheiden.

Zu den niederen Programmiersprachen gehören vor allem die Maschinensprachen der jeweiligen Rechner. Sie sind im allgemeinen prozessorspezifisch¹. Sie sind für den Rechner für den sie entwickelt wurden direkt und problemlos, für den Menschen aber vergleichsweise schwer verständlich. Sie bestehen letztlich aus ziemlich abstrakten Zahlenfolgen. Für Menschen ist es zwar möglich in einer Maschinensprache zu programmieren, die Arbeitsgeschwindigkeit bei der Programmerstellung ist bei dieser Vorgehensweise jedoch unakzeptabel niedrig und die Fehleranfälligkeit

Programmierung in Maschinensprache:

Zeitaufwand und Fehleranfälligkeit sind hoch.

¹ Der Prozessor ist derjenige Teil Ihres Rechners, der die eigentliche „Rechenarbeit“ leistet.

keit in der Regel sehr hoch. Aus diesem Grund wurde schon früh Möglichkeiten gesucht, um die Programmierarbeit wirtschaftlicher gestalten. Man „erfand“ die Programmiersprachen.

In einem ersten Schritt entstanden die noch sehr maschinennahen Assemblersprachen. Üblicherweise wird bei diesen Sprachen der Operationscode des Befehls durch einen mnemotechnischen Buchstabencode (z. B. ADD statt 100_{16} für Addition). Operanden können wahlweise durch Zahlen oder alphanumerische Symbole dargestellt werden.

Aus verschiedenen Gründen und vor verschiedenen fachlichen und geschäftspolitischen Hintergründen entstanden in den letzten 50 Jahren zahlreiche höhere Programmiersprachen. Die meisten von ihnen basieren auf einer Art mathematischer Notation die mit ein paar englischen Vokabeln kombiniert ist. Der folgende Text zeigt einen Ausschnitt aus einem Programm in der höheren Programmiersprache Pascal:

```
if a >= b then
  c := 1
else
  c := 2;
```

Dieses kurze Programmstück sagt aus, dass falls der Wert von *a* größer oder gleich dem Wert von *b* ist, der Variablen *c* der Wert 1 zugewiesen wird. Andernfalls bekommt *c* den Wert 2.

1.2.2. Syntax und Semantik

Als Syntax bezeichnet man die Lehre vom Satzbau. So wie in einer natürlichen Sprache aus dem verfügbaren Wortschatz nach bestimmten Regeln korrekte Sätze gebildet werden, werden auch die Sätze (Ausdrücke, Anweisungen...) eines Programms nach festen Regeln gebildet. Die Syntaxregeln einer Programmiersprache können mit Methoden wie der Backus-Naur-Form (BNF) oder Syntaxdiagrammen wie sie in diesem Buch vorzugsweise zur Anwendung kommen dargestellt werden.

Die Semantik beschäftigt sich mit der Bedeutung der Sätze. Syntaktisch korrekte Sätze können semantisch falsch sein. Diese semantischen Fehler können zu einem gewissen Teil vom Compiler entdeckt werden. Sie dürfen nicht mit logischen Fehlern verwechselt werden.

Beispiel:

```
var a, b, c: integer;
.....
.....
a := b or c;
```

Der Programmausschnitt ist zwar syntaktisch korrekt, aber semantisch fehlerhaft, da mit `or` nur Daten vom Typ `boolean` verknüpft werden dürfen.

In natürlicher Sprache ist das vergleichbar mit den Sätzen

- Herr Maier isst sein Auto.

und

- Herr Maier wäscht sein Auto.

Beide Sätze sind syntaktisch korrekt. Semantisch korrekt ist nur der letztere.

1.2.3. Vom Compiler zur Integrierten Entwicklungsumgebung

Die Texte, die der Programmierer² in dieser Sprache erstellt, kann der Rechner nicht direkt verstehen. Sie müssen für ihn in seine Maschinensprache übersetzt werden. Diese Übersetzung kann vor oder während der Ausführung des Programms erfolgen.

Erfolgt die Übersetzung vor der Programmausführung, so spricht man von einer Compilierung, das Programm, das die Übersetzung ausführt ist der Compiler.

Die Übersetzung während der Programmausführung bezeichnet man als Interpretation und das dafür eingesetzte Programm entsprechend als Interpreter.

In der Softwareindustrie werden heute sowohl interpretierbare Sprachen als auch compilierbare Sprachen eingesetzt. Compilierbare Sprachen erzielen wesentlich besseres Laufzeitverhalten als interpretierbare. Bei gleicher Aufgabenstellung und gleichem Rechner sind die Laufzeiten von compilierten Programmen kürzer als von interpretierten, da die zeitaufwändige Übersetzung bereits vor der Ausführung erfolgte.

Meist besteht ein Programm aus mehreren Teilen (Modulen). Die Module, die der Programmierer in der (höheren) Programmiersprache - z. B. Pascal - schreibt, nennt man Quellmodule. Werden sie übersetzt, so entsteht daraus die Objektmodule. Die Verknüpfung der Objektmodule durch den Binder (Linker) führt dann zum lauffähigen Programm (Laufzeitprogramm, Laufzeitsystem, Executable).

Editor, Compiler, Binder und Debugger werden heute üblicherweise innerhalb eines einzigen Programms betrieben. Dieses wird als integrierte Entwicklungsumgebung

Compilierung vor der Programmausführung, **Interpretation** während der Programmausführung.

Quellcode → Objektcode
(Compiler)

Objektcode → Ausführbares Programm
(Linker)

² Alles was dieses Buch über den Umgang mit dem Entwicklungssystem Lazarus darstellt gilt selbstverständlich gleichermaßen für männliche und weibliche Nutzer. Der übersichtlicheren Darstellung wegen wird ab sofort nur noch die männliche Form verwendet.

(IEU, engl. Integrated Development Environment (IDE)) bezeichnet. Lazarus ist ein Beispiel für eine solche Integrierte Entwicklungsumgebung. Weitere IEUs sind z. B. Microsoft Visual Studio, Embarcadero RADs, Qt oder Java Swing.

1.3. Pascal – Ein kurzer historischer Überblick

1.3.1. Der Anfang

Pascal ist eine Weiterentwicklung von Algol 60 einer besonders in Europa populären frühen höheren Programmiersprache. Die Sprache Pascal wurde von Niklaus Wirth an der Eidgenössischen Technischen Hochschule in Zürich vor allem in Hinblick auf die Verbesserung der Programmierausbildung entwickelt.

Pascal zwingt zu einem klaren Umgang mit Daten- und Programmstrukturen und fördert auf diese Weise die Erstellung von wohl strukturiertem und gut verständlichem Code

1.3.2. Wichtige Implementationen

Sehr große Verbreitung in der professionellen Programmierung fand Pascal als Borland/Turbo Pascal (später Delphi). Dabei handelt es sich um gegenüber dem Ur-Pascal wesentlich erweiterte und verbesserte Versionen. Diese bieten neben dem ursprünglichen Sprachumfang von Pascal einen einfachen Zugriff auf wichtige Funktionen des Betriebssystems. Später kamen dann noch objektorientierte Elemente hinzu.

1.3.3. Free Pascal und Lazarus

[Free Pascal](#) ist ein unter der GNU Lesser General Public License stehendes Open-Source-Projekt, das sich zum Ziel gesetzt hat, einen freien 32 bzw. 64-Bit-Compiler zu erstellen, der 100-prozentig kompatibel zu Turbo Pascal und Delphi sein soll und mittlerweile eine leichte Portierung von Pascal-Programmen auf fast alle gängigen Betriebssysteme und Hardwareplattformen ermöglicht.

Lazarus ist eine Integrierte Entwicklungsumgebung (IEU, engl. IDE für Integrated Development Environment) für Free Pascal, die auch verschiedene Komponenten zur Verfügung stellt. Die IDE ist sehr Delphi-ähnlich gestaltet und verwendet unter Unix das GTK+³ als Grafik-Toolkit, unter Windows (Win32/Win64/Windows CE)

3 GTK+ = **GTK+** (*GIMP Toolkit*) ist eine freie Komponentenbibliothek unter der LGPL (Lesser Gnu Public License), mit der grafische Benutzeroberflächen (GUI) für Software erstellt werden können.

setzt es auf dem nativen⁴ API⁵ auf, und auf Apple-Betriebssystemen kann wahlweise das native Carbon-API oder das X-Window-System verwendet werden. Darüber hinaus unterstützt Lazarus Cross Compiling, so dass auch Software für weitere Plattformen wie Windows CE, OS/2, Palm OS oder Nintendo DS entwickelt werden kann. Lazarus ist noch nicht fertiggestellt. In kurzen Abständen werden aber neue Betaversionen veröffentlicht, die – je nach Plattform in unterschiedlichem Maße – einen wachsenden Teil der geplanten Funktionalität implementieren. [LAZHOME]

1.4. Didaktische Ausrichtung

Grundsätzlich ist dieses Buch als Programmierlehrbuch nach dem Prinzip „Learning by Doing“ angelegt. Das Erlernen der Programmiersprache erfolgt dabei durch Erarbeiten von Programmlösungen. Seine Gliederung orientiert sich an den zu erwerbenden Programmierkenntnissen. Diese werden von Kapitel zu Kapitel durch aufeinander aufbauende Programmieraufgaben erweitert. D. h. von Aufgabe zu Aufgabe werden die Kenntnisse vertieft. Gleichzeitig dringen Sie von Kapitel zu Kapitel tiefer in die Möglichkeiten der Entwicklungsumgebung Lazarus ein.

Die Kapitel dieses Lehrbuchs sind durchgängig in folgende vier Abschnitte gegliedert:

- Aufgabenstellung: Welche Aufgabe soll das Programm erfüllen? Wie soll es eingesetzt werden?
- Beitrag von Pascal: In diesem Abschnitt wird dargestellt, welche (Free-) Pascal-Sprachelemente Sie zur Lösung der Aufgabe benötigen und was Sie bei ihrer Benutzung beachten müssen. Sowohl für Pascal als auch für Lazarus (nächster Absatz) werden im allgemeinen nur solche Elemente aufgeführt die in vorangegangenen Kapiteln noch nicht besprochen wurden
- Beitrag von Lazarus: Hier erfahren Sie, welche Teile der Entwicklungsumgebung Lazarus Sie zur Programmentwicklung einsetzen müssen und welche Hilfe Sie von diesen erwarten können.
- Lösung: Vorstellung und Erläuterung des kompletten Programmtexts.
- Weitere Aufgaben, ggf. mit Lösungen.

4 Nativ von engl. native (=eingeboren) bezeichnet eine Entwicklung oder ein Subsystem, die/das vom Grundsatz her zu einer bestimmten Arbeitsumgebung (Hardware und/oder betriebssystem) gehört.

5 API: Application Programming Interface, Schnittstelle eines Programms A (häufig so wie auch hier eines Betriebssystems), die es ermöglicht die Fähigkeiten dieses Programms A durch ein Programm B zu nutzen.

Wer ein Lehrbuch wie dieses schreibt, steht manchmal vor dem Problem von Ei und Henne, d. h., dass zur Erklärung des momentan erarbeiteten Stoffs idealerweise Elemente erforderlich wären, die erst in späteren Kapiteln erarbeitet werden. Um Sprünge im Text zu vermeiden wird in diesem Fall folgendermaßen vorgegangen:

- Die gestellte Aufgabe wird zunächst mit den bis zum jeweiligen Kapitel vorgestellten verfügbaren Sprachmitteln gelöst. Das führt zu verständlichen, funktionierenden, aber nicht optimalen und vielleicht etwas umständlichen Lösungen.
- Wenn in späteren Kapiteln zusätzliches Wissen erarbeitet wurde, das elegantere Lösungen ermöglicht, wird die Aufgabenstellung nochmals aufgenommen. Ein Beispiel dafür ist der Richtigkeitsnachweis für die DeMorganschen Regeln, der erstmals in 4.3.2 auftritt und für den dann in 11.4.1.1 eine wesentlich elegantere Lösung gezeigt wird.

1.5. Wenn Sie nicht Seite für Seite vorwärts arbeiten wollen.....

Dieses Buch wendet sich sowohl an absolute Anfänger als auch an ein breites Spektrum von Quereinsteigern.

Anfängern ohne Vorkenntnisse empfehle ich, dieses Buch von Anfang an Seite für Seite gründlich durchzuarbeiten.

Wenn Sie schon Pascal-Grundkenntnisse besitzen und vor allem an einer Einarbeitung in die Windows-Programmierung interessiert sind, werden Sie nach einem kurzen Studium von Kapitel 1 wahrscheinlich vor allem die Kapitel 9, 15 bis 18 intensiv durcharbeiten und die übrigen eher zum Nachschlagen nutzen.

Wer sich speziell für die Grafikprogrammierung interessiert wird sich vor allem mit den Kapiteln 16 und 17 befassen.

Wenn Sie eine Lösungshilfe für ein bestimmtes Anwendungsgebiet suchen, dann sollten Sie im Stichwortverzeichnis unter **Anwendungsthemen** nachsehen.

2. „Hallo Welt“ - Ihr erstes Programm

2.1. Aufgabe: Erstellen Sie eine (Konsol-) Anwendung

In diesem Kapitel erfahren Sie, wie Sie mittels Lazarus ein minimales Programm erstellen können. Das Programm soll ganz einfach den Text "Hallo Welt" auf dem Bildschirm des Rechners ausgeben.

Dabei verzichten Sie zunächst auf eine grafische Benutzeroberfläche (engl. Graphical User Interface (GUI)) auch Windows oder Fensteroberfläche genannt, sondern Sie beschränken sich zunächst auf eine Anwendung mit alphanumerischer Benutzeroberfläche bei der die Ausgabe Zeichen für Zeichen direkt auf dem Bildschirm erfolgt, eine so genannte Konsolanwendung.

Zunächst werden nur Konsolanwendungen erstellt.

2.2. Der Beitrag von Pascal

2.2.1. Aufbau eines Programms

Ihre Programmieraufgabe besteht in der Erstellung eines einzelnen Programms. Grundsätzlich werden Programme durch das Schlüsselwort `program` auf das der Name des Programms folgt eingeleitet. Den Kern des Programms, den so genannten Anweisungsteil, in dem niedergeschrieben wird, was das Programm tun soll, leitet das Schlüsselwort `begin` ein und das Schlüsselwort `end` auf das ein Punkt folgt schließt ihn ab (Syntaxdiagramm 2.1).

Ein Schlüsselwort in Pascal ist eine vorgeschriebene Zeichenfolge die innerhalb des Programms nur in einer genau festgelegten Bedeutung verwendet werden darf. So darf z. B. der Bezeichner `program` z. B. ausschließlich zur Definition des Programmanfangs verwendet werden. In den Tabellen 2.1 bis 2.3 wird eine Übersicht über die Pascal-Schlüsselwörter gegeben. Der genaue Umfang der Schlüsselwörter hängt vom eingestellten Modus ab.

Ein Schlüsselwort hat eine genau festgelegte Bedeutung. Nur in dieser darf es verwendet werden.

Für den Namen (Bezeichner) des Programms gelten - wie für alle anderen im Programm auftretenden Bezeichner auch - die folgenden Regeln (Syntaxdiagramm 2.2):

- Der Bezeichner muss aus Buchstaben (A . . . Z, a . . . z, underscore (`_`) oder Dezimaliffern bestehen.
- Er muss mit einem Buchstaben oder underscore beginnen.

Zwischen groß und klein geschriebenen Zeichen wird nicht unterschieden.

- In Pascal wird nicht zwischen Groß- und Kleinbuchstaben unterschieden. In anderen Programmiersprachen (z. B. in C oder C++) ist das durchaus der Fall. D. h. die Zeichenfolgen `MeinProgramm` und `MEINPROGRAMM` werden als identische Bezeichner behandelt.
- Die Länge des Bezeichners ist grundsätzlich beliebig. In besonderen Implementations- oder Anwendungsfällen können aber Längenbeschränkungen vorgesehen sein.

<i>absolute</i>	<i>and</i>	<i>array</i>	<i>asm</i>	<i>begin</i>
<i>case</i>	<i>const</i>	<i>constructor</i>	<i>destructor</i>	<i>div</i>
<i>do</i>	<i>downto</i>	<i>else</i>	<i>end</i>	<i>file</i>
<i>for</i>	<i>function</i>	<i>Goto</i>	<i>if</i>	<i>implementation</i>
<i>in</i>	<i>inherited</i>	<i>inline</i>	<i>interface</i>	<i>label</i>
<i>mod</i>	<i>nil</i>	<i>not</i>	<i>object</i>	<i>of</i>
<i>on</i>	<i>operator</i>	<i>or</i>	<i>packed</i>	<i>procedure</i>
<i>program</i>	<i>record</i>	<i>reintroduce</i>	<i>repeat</i>	<i>self</i>
<i>set</i>	<i>shl</i>	<i>shr</i>	<i>string</i>	<i>then</i>
<i>to</i>	<i>type</i>	<i>unit</i>	<i>until</i>	<i>uses</i>
<i>var</i>	<i>while</i>	<i>with</i>	<i>xor</i>	

Tabelle 2.1: Liste der Pascal-Schlüsselwörter (Turbo-Pascal-Modus)

<i>dispose</i>	<i>false</i>	<i>true</i>	<i>exit</i>	<i>new</i>
----------------	--------------	-------------	-------------	------------

Tabelle 2.2: Liste der zusätzlichen Schlüsselwörter (Free-Pascal-Modus)

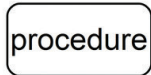
<i>as</i>	<i>class</i>	<i>dispinterface</i>	<i>except</i>	<i>exports</i>
<i>finalization</i>	<i>finally</i>	<i>initialization</i>	<i>inline</i>	<i>is</i>
<i>library</i>	<i>on</i>	<i>out</i>	<i>packed</i>	<i>property</i>
<i>raise</i>	<i>resourcestring</i>	<i>threadvar</i>	<i>try</i>	

Tabelle 2.3: Liste der zusätzlichen Schlüsselwörter (Object-Pascal-Modus)

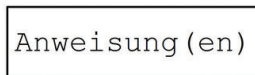
Programm

Bezeichnung des durch das Syntaxdiagramm definierten Sprachelements

Grundsymbol (Terminalsymbol), aus einem Zeichen bestehend



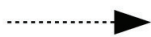
Grundsymbol (Terminalsymbol), aus mehreren Zeichen bestehend



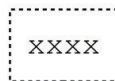
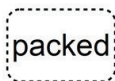
Zusammengesetztes Symbol (Nonterminalsymbol), wird in der Regel durch ein eigenes Syntaxdiagramm dargestellt.



Flusslinie. Durch Verfolgung der Flusslinien ergibt sich der mögliche Aufbau des beschriebenen Sprachelements.



Flusslinie. In diesem Bereich ist das Sprachelement vereinfacht dargestellt



Das Element wird der Vollständigkeit halber aufgeführt aber im Text nicht besprochen.

Abbildung 2.1: Darstellungselemente der Syntaxdiagramme

Diese Regeln begegnen uns in ähnlicher Form in jeder Programmiersprache als Regeln zur Bildung von Bezeichnern (Namen) (Syntaxdiagramm 2.2). Das Grundmuster für ein Programm hat demnach das folgende Bild:

```
program Name_Des_Programms;
...
begin
...
end.
```

2.2.2. Syntaxdiagramme (Fahrdiagramme)

Bei der Anwendung von Programmiersprachen müssen genau wie bei natürlichen Sprachen strenge Regeln eingehalten werden, da sie eine genau definierte Syntax (Grammatik) besitzen. Zur Beschreibung der Syntax kann man freie Prosa- oder auch formale Texte (z. B. die Backus-Naur-Form (BNF)) benutzen. Besonders anschaulich sind jedoch Syntaxdiagramme (bisweilen auch Fahrdiagramme genannt). Die 4 bzw. 5 Elemente dieser Diagramme sind in Abbildung 2.1 dargestellt. Mit ihrer Hilfe können neu eingeführte Sprachelemente durch Grundsymbole und bereits zuvor definierte Sprachelemente beschrieben werden.

- Terminalsymbole sind Grundsymbole der Programmiersprache, die nicht weiter aufgeteilt werden können. Je nach der Länge des eingetragenen Textes werden sie als Kreise oder Rechtecke mit abgerundeten Ecken dargestellt. Terminalsym-

bole können Einzelzeichen (Buchstaben Dezimalziffern, gewisse Sonderzeichen (z. B. +, -, *...), Zeichen kombinationen (z. B. (/) oder Schlüsselwörter sein. Für die Beschriftung der Terminalsymbole wird eine serifenlose Schrift (Arial) verwendet.

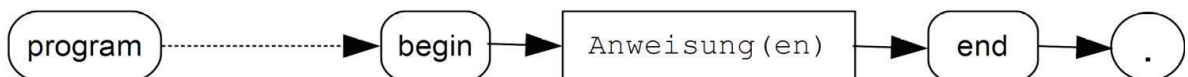
- Nichtterminalsymbole (zusammengesetzte Symbole) sind Symbole, die nach bestimmten Regeln aus Nichtterminalsymbolen und/oder Terminalsymbolen gebildet werden. Für die Beschriftung der Nichtterminalsymbole wird eine Serifenschrift (Courier New) verwendet.
- Flusslinien stellen dar, wie die Symbole aus denen ein Wort besteht aufeinander folgen. Flusslinien können nur in einer Richtung durchlaufen werden. An Verzweigungen kann eine beliebige Richtung eingeschlagen werden.
- Dort wo die Syntax aus didaktischen Gründen vorläufig unvollständig dargestellt werden gestrichelte (— — — —) Flusslinien eingesetzt.

Syntaxdiagramme werden nur im unbedingt notwendigen Umfang eingeführt.

Um den Umfang des Buches nicht zu sehr aufzublähen werden Syntaxdiagramme nur in dem Maße gebracht, wie sie zum Verständnis von Lazarus erforderlich sind. Auf eine vollständige Syntaxdarstellung in Form von Syntaxdiagrammen wird verzichtet.

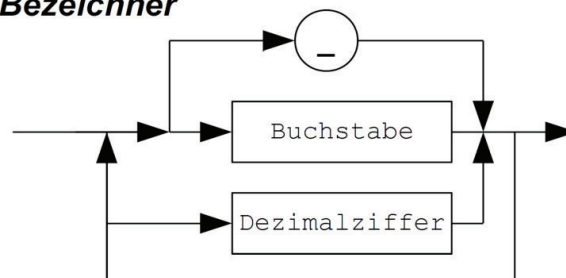
Die wesentlichen bis jetzt eingeführten Sprachelemente und Konstrukte von Pascal sind nachstehend als Syntaxdiagramme dargestellt:

Programm

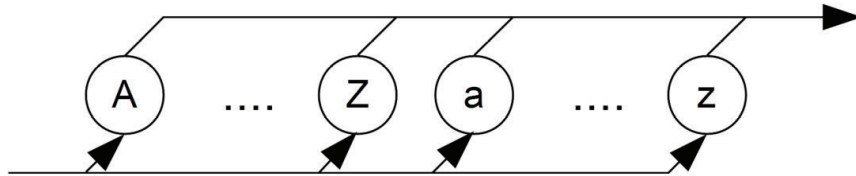


Syntaxdiagramm 2.1: Programm (vereinfacht)

Bezeichner



Syntaxdiagramm 2.2: Bezeichner

Buchstabe

Syntaxdiagramm 2.3: Buchstabe

2.2.3. Darstellung von Texten

Texte, genauer Textkonstante, werden in Pascal durch eine beliebige, der Aufgabenstellung entsprechende Zeichenfolge, die durch Hochkommata (') begrenzt ist dargestellt. Das Hochkomma finden Sie im rechten Teil Ihrer Tastatur auf der selben Taste wie das #⁶.



Das Hochkomma finden Sie auf der selben Taste wie das #

Wie stelle ich ein Hochkomma dar?

Wie bisher besprochen wird Hochkomma zur Begrenzung von Texten in Textkonstanten verwendet. Es ist also selbst nicht Bestandteil des Texts. Was ist zu tun, wenn ein Hochkomma im Text benötigt wird? Die Lösung ist einfach: Wenn in einer Textkonstanten ein Hochkomma auftreten soll, so sind stattdessen zwei Hochkommata zu verwenden.

Unseren Text *Hallo Welt* schreiben wir im Programm in der Form 'Hallo Welt'. An die Stelle von *Onkel Tom's Hütte* tritt 'Onkel Tom''s Hütte'.

Ein Hochkomma, das geschrieben werden soll wird in der Anweisung des Programms verdoppelt.

2.2.4. Ausgabeanweisung

Die Datenausgabe kann in Pascal mit den Anweisungen `Write(Liste)` und `WriteLn (Liste)` erfolgen⁷. Die Liste kann sehr viele verschiedene Formen annehmen (...). Im einfachsten Fall besteht sie lediglich aus einer Konstanten. Weitere Erläuterungen zur Parameterliste bei der Pascal Ein-/Ausgabe finden Sie in Kapitel Fehler: Referenz nicht gefunden. Das Schreiben beginnt bei beiden Anweisungen am aktuellen Ort des Bildschirmcursors.

Bei `Write` werden genau die in den Listenelementen enthaltenen Zeichen ausgegeben. Bei diesen Zeichen kann es sich um so genannte darstellbare Zeichen (z. B. A bis Z und a bis z aber auch + - () u. ä.) oder auch um nicht darstellbare Zeichen (Steuerzeichen wie z. B. den Zeilenvorschub oder auch ein akustisches Signal) handeln. Bei `WriteLn` ist die Sache ähnlich. Nur wird dort an die auszudruckenden

⁶ Gilt für Deutsche Tastaturen

⁷ Die ausführliche Behandlung der Ein- und Ausgabe erfolgt in Kapitel Fehler: Referenz nicht gefunden



Abbildung 2.2: Zweimalige Anwendung von `Write(...)`;

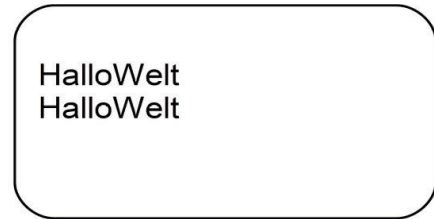


Abbildung 2.3: Zweimalige Anwendung von `WriteLn(...)`;

Zeichen ein Wagenrücklauf⁸ und ein Zeilenvorschub angefügt. Nach der Ausführung von `WriteLn` steht der Cursor auf Spalte 1 der Folgezeile.

Nach zweimaliger Ausführung von

```
Write('Hallo Welt');
```

ergibt sich die folgende Bildschirmansicht gemäß Abbildung 2.2.

Entsprechend ergibt sich nach zweimaliger Ausführung von

```
WriteLn ('Hallo Welt');
```

eine Bildschirmansicht gemäß Abbildung 2.3.

2.2.5. Elementare Eingabebezeichnung

Wenn alles genau so funktionieren würde, wie Sie es nach der Aufgabenstellung wahrscheinlich erwarten, brauchten Sie sich mit den Informationen dieses Abschnitts gar nicht zu befassen. Da das Programm jedoch einen zwar erklärbaren aber wahrscheinlich für den Anfänger unerwarteten Effekt zeigen wird - das Fenster in dem die Anwendung läuft, schließt sich bereits Sekundenbruchteile nach dem Programmstart wieder - benötigen Sie schon hier die Möglichkeit, auf die Betätigung einer Eingabetaste (hier der Enter-Taste) zu warten.

In einfachster Weise tun Sie das mit der Eingabebezeichnung `ReadLn`. Bei Ausführung wartet das Programm auf eine Eingabe in Form der Betätigung der Enter-Taste. Nach der Betätigung dieser Taste wird der Schreibcursor auf die Spalte 1 der Folgezeile gesetzt und die Programmausführung wird mit der auf `ReadLn` folgenden Anweisung fortgesetzt.

⁸ Diese heute noch übliche Bezeichnung ist historisch zu verstehen, da früher die Positionieren und auf den Zeilenanfang durch Verschieben des "Wagens" des Ausgabegeräts (Fernschreiber, elektrische Schreibmaschine o. ä.) erfolgte, auf dem das Papier eingespannt wurde