E. Hering/K. Scheurer

Fortgeschrittene Programmiertechniken in Turbo Pascal

Programmieren von Mikrocomputern

Die Bände dieser Reihe geben den Benutzern von Heimcomputern, Hobbycomputern bzw. Personalcomputern über die Betriebsanleitung hinaus zusätzliche Anwendungshilfen. Der Leser findet wertvolle Informationen und Hinweise mit Beispielen zur optimalen Ausnutzung seines Gerätes, besonders auch im Hinblick auf die Entwicklung eigener Programme.

Bisher erschienene Bände

Band 1 **Einführung in BASIC**
von W. Schneider

Band 3 **BASIC für Fortgeschrittene**
von W. Schneider

Band 4 **Einführung in Pascal**
von W. Schneider

Band 6 **BASIC-Programmierbuch zu den grundlegenden Ablaufstrukturen der Datenverarbeitung**
von E. Kaier

Band 7 **Lehr- und Übungsbuch für Commodore-Volkscomputer**
von G. Oetzmann

Band 9 **Einführung in die Anwendung des Betriebssystems CP/M**
von W. Schneider

Band 10 **Datenstrukturen in Pascal und BASIC**
von D. Herrmann

Band 11 **Programmierprinzipien in BASIC und Pascal**
von D. Herrmann

Band 12 **Assembler-Programmierung von Mikroprozessoren (8080, 8085, Z 80) mit dem ZX Spectrum**
von P. Kahlig

Band 13 **Strukturiertes Programmieren in BASIC**
von W. Schneider

Band 14 **Logo-Programmierkurs für Commodore 64 Logo und Terrapin Logo (Apple II)**
von B. Schuppar

Band 15 **Entwerfen von Programmen (Commodore 64)**
von G. Oetzmann

Band 16 **Einführung in die Anwendung des Betriebssystems MS-DOS**
von W. Schneider

Band 17 **Einführung in die Anwendung des UCSD p-Systems**
von K. Buckner/M. J. Cookson/ A. I. Hinxman/A. Tate

Band 18 **Mikrocomputer-COBOL**
von W. Kähler

Band 19 **Fortgeschrittene Programmiertechniken in Turbo Pascal**
von E. Hering und K. Scheurer

Band 20 **Einführung in die Anwendung des Betriebssystems Apple DOS (Apple II)**
von H. R. Behrendt und H. Junghans

Band 21 **LOGO? LOGO!**
von K. Haussmann

Band 22 **Einführung in Turbo Pascal unter CP/M 80**
von G. Harbeck

Band 23 **Pascal mit der Turtle**
von K. und K. H. Beelich

Band 24 **Programmieren mit UNIX**
von G. Martin und M. Trostmann

Band 25 **Murmeltierwelt und Pascal**
von H. Pinke

Programmieren von Mikrocomputern Band 19

Ekbert Hering
Karl Scheurer

Fortgeschrittene Programmiertechniken in Turbo Pascal

Springer Fachmedien Wiesbaden GmbH

1986

Ursprünglich erschienen bei Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig 1986.

Additional material to this book can be downloaded from http://extras.springer.com.

ISBN 978-3-528-04467-1 ISBN 978-3-663-06857-0 (eBook)
DOI 10.1007/978-3-663-06857-0

Vorwort

Während sich die Besitzer von Homecomputern meist nur mit überschaubaren Problemen beschäftigen, müssen die Programmierer kommerziell anwendbarer Systeme überwiegend komplexe Fragestellungen bewältigen. Eine bewährte Methode, solche Aufgaben zu lösen besteht darin, diese in einfacher zu lösende Teilprobleme zu zerlegen. Eine weitere Steigerung der Produktivität ist möglich, wenn Teilbereiche, die sich wiederholen, zusammengefaßt werden. Die einzelnen Teilprobleme und ihre programmtechnischen Lösungen werden standardisiert und in Programmbibliotheken abgelegt. Durch deren Verwendung wird es möglich, komplexe Probleme mit vertretbarem Zeitaufwand wirtschaftlich zu lösen.

Leider sind viele in der EDV-Ausbildung Lehrende vorwiegend immer noch der Ansicht, daß es genügt, ihren Schülern die Fähigkeiten eines „Homecomputer-Programmierers" beizubringen. Aus „pädagogischen Gründen" erhalten die Schüler vereinfachte oder praktisch irrelevante Problemstellungen mit dem Ergebnis, daß den meisten Absolventen derartiger Lehrgänge das Verständnis für die Notwendigkeit einer Systematik in der Programmierung abgeht. Daß dies heute noch üblich ist, erfuhr einer der Verfasser, als er für seine Firma auf einen Lehrgang zur „Erweiterung seiner Programmierkenntnisse" geschickt wurde.

Neben der Tatsache, daß er bei dieser Gelegenheit zum x-ten mal das „Bubble"-Sortverfahren (das ungünstigste von allen möglichen Sortierverfahren) neu erfinden sollte, wurde kein gangbarer Weg aufgezeichnet, wie die neu erworbenen Kenntnisse die Probleme der Praxis besser lösen helfen. Diese am eigenen Leib und von anderen Kursteilnehmern in gleicher Weise empfundene Erfahrung war einer der Hauptgründe, dieses Buch zu schreiben. Das erste Anliegen dieses Buches ist es, dem Leser die Erstellung von Standardroutinen zu demonstrieren, d. h. Prozeduren und Funktionen zu entwickeln, die häufig auftretende Problemstellungen in sauberer, reproduzierbarer Weise bewältigen. Um dies zu erreichen, verzichten wir darauf, allgemeine, praxisfremde Probleme zu behandeln, sondern beschreiben vielmehr Routinen aus unserer Praxis, von denen wir aus Erfahrung wissen, daß sie nützlich sind. Jede Prozedur und jede Funktion wurde in der hier vorliegenden Form auf mindestens zwei verschiedenen Rechnern (Tandy Modell 1000, Siemens PC-D) gründlich ausgetestet.

Das zweite Anliegen dieses Buches ist es, die Routinen gut zu schreiben. Dazu wollen wir dem Leser folgende Prinzipien vermitteln:

1. klarer Entwurf:
   Er ermöglicht eine leichte Pflege und Änderung bestehender Programme.
2. Benutzerfreundlichkeit:
   Sie ermöglicht den gewünschten Anforderungskomfort.
3. Zuverlässigkeit:
   Sie sichert die Richtigkeit der Ergebnisse unter allen gegebenen Randbedingungen.
4. Effizienz:
   Sie ist notwendig, damit sich der Einsatz dieser Routinen auch lohnt.

Es gibt mittlerweile eine Unzahl von Schlagwörtern die sich mit der Verbesserung des Programmiervorgangs beschäftigen. Die wichtigsten davon sind:

1. strukturiertes Programmieren
2. „Top-down"-Entwurf
3. strukturierter Entwurf

Das strukturierte Programmieren umfaßt im engeren Sinne das Programmieren mit einer beschränkten Anzahl von Kontrollflußanweisungen und der Vermeidung von GOTOs.

Der „Top-down"-Entwurf beschreibt ein Programmierproblem zunächst in übergeordneten Einheiten, um diese von Stufe zu Stufe immer präziser zu gestalten, bis das Programm fertiggestellt ist.

Beim strukturierten Entwurf wird ein Gesamtsystem aus fugenlos zusammenpassenden Teilen aufgebaut. Diese Teile sind soweit eigenständig, daß sie unabhängig voneinander modifiziert werden können.

Obwohl jede dieser Methoden die Produktivität des Programmierers und die Qualität der Programme erheblich verbessern kann, ist es dennoch gefährlich, darauf zu vertrauen, daß die blinde Anwendung irgendeiner speziellen Technik automatisch zu guten Programmen führt. Wir glauben nicht, daß gutes Programmieren durch Befolgen von abstrakten Programmierprinzipien erlernt werden kann. Auch ist es wenig sinnvoll, sich mit künstlichen oder praxisfernen Problemen zu befassen. Anstatt abstrakte Konzepte wie strukturiertes Programmieren oder „Top-Down"-Entwurf nur theoretisch abzuhandeln, versuchen wir, die unserer Meinung nach wichtigen Bestandteile herauszufiltern und in unseren Routinen einzusetzen. Dadurch sollte ihre Bedeutung bei der Lösung realer Probleme und ihr praktischer Nutzen besser erkennbar werden.

Unsere Routinen vermeiden den Einsatz von GOTOs weder aus Respekt vor irgendwelchen theoretischen Prinzipien, noch in der Hoffnung, dadurch automatisch als gut anerkannt zu werden, sondern vielmehr aus der Erfahrung, daß zumindest in Pascal der Änderungsaufwand (Labeldeklaration, GOTO Anweisung und Label) den möglichen Effizienzgewinn dieser Anweisung bei weitem übersteigt.

Der Einsatz des „Top-Down"-Entwurfs beruht auf der persönlichen Erfahrung, daß damit schneller und fehlerfreier programmiert werden kann. Die Beachtung des strukturierten Entwurfes ist für den Einsatz von Standardroutinen obligatorisch, da sonst die Zuverlässigkeit eines Gesamtsystems durch Wechselwirkung zwischen Einzelkomponenten unzulässig eingeschränkt wird.

Anstatt nur Problemstellung und Endprodukt zu präsentieren, beschreiben wir, wie wir bei der Programmierung vorgegangen sind. Auch wenn der Leser nicht der Ansicht ist, unsere Art zu programmieren sei die bestmögliche, sollte ihm die Auseinandersetzung mit unserer Vorgehensweise, bzw. mit den Überlegungen, die uns beim Entwurf und bei der Implementierung beeinflußt haben, genügend Anregungen für eigene Entwicklungen liefern.

Dieses Buch bietet eine Fülle von Standardroutinen. Wir haben uns dabei auf Probleme konzentriert, mit denen der Leser wahrscheinlich sehr häufig konfrontiert werden wird. Aus Platzgründen haben wir uns auf einen zentralen Bereich der Datenverarbeitung, dem Erkennen und Umformen von Zeichenfolgen und Mustern, konzentriert. Obwohl diese

Fragestellung den Bereich der numerischen Datenverarbeitung nicht berühren muß, haben wir einen Abschnitt der Verarbeitung von Matrizen gewidmet.

Wir haben uns dabei nicht bei der Erläuterung der eingesetzten Rechenverfahren aufgehalten, sondern vielmehr mit den Möglichkeiten beschäftigt, wie Daten, die mehr als eine Dimension zur Beschreibung benötigen, effizient dargestellt und verarbeitet werden können.

Unabhängig von der Anwendung ist die Wahl einer guten Programmiersprache wichtig, damit die Darstellung nicht durch unnötige Details verkompliziert wird. Nach dieser Forderung bleiben nur noch drei Sprachen übrig: BASIC, Pascal und C.

Von diesen scheidet zuerst die Sprache BASIC aus. Dies geschieht nicht etwa, weil es nicht schick ist, in BASIC zu programmieren, sondern vielmehr, weil die im Moment gängigen BASIC-Dialekte keine saubere Möglichkeit bieten, separate Unterprogramme zu schreiben. Da im Moment Bestrebungen im Gange sind, BASIC zu standardisieren, könnte dieser Nachteil wegfallen.

Obwohl C die beste Möglichkeit bietet, standardisierte Routinen zu entwickeln, scheidet es dennoch aus, da die im Moment auf PCs erhältlichen C-Compiler eine unzureichende Programmierumgebung beinhalten. Sollte dies in absehbarer Zeit verbessert werden, so muß C durchaus beachtet werden.

Die Entscheidung für Pascal wurde eigentlich nur deshalb gefällt, weil seit 1984 mit Turbo Pascal eine effiziente Sprache vorliegt, die gleichzeitig einen bis dahin auf den meisten Systemen unbekannten Programmierkomfort bietet. Da dieser Pascal-Dialekt neben wichtigen Erweiterungen eine ganze Reihe von empfindlichen Schwachstellen des Standard Pascal behebt und dazu noch konkurrenzlos preiswert ist, haben wir unsere Routinen in dieser Sprache geschrieben.

Aalen, März 1986

Ekbert Hering
Karl Scheurer

Inhaltsverzeichnis

1 Fortgeschrittene Programmiertechniken in Turbo Pascal 1

1.1 Vergleich Standard Pascal und Turbo Pascal 1
1.2 Unterschiede von Standard Pascal zu Turbo Pascal 3
1.2.1 Vereinbarungsfolge 3
1.2.2 Typdeklaration 4
1.2.3 Existenzbereich von Variablen 4
1.2.4 Pointer 5
1.2.5 Parameterübergabe an Prozeduren und Funktionen 5
1.3 Turbo Pascal und Software-Engineering 5

2 Anpassungsfreie Unterprogramme durch typfreie Parameter 8

2.1 Datentypen und Datenstrukturen 8
2.2 Analogie zwischen numerischen und nichtnumerischen Daten 10
2.3 Mustererkennung 11
2.3.1 Klassifizieren von Zeichen 12
2.3.2 Vergleich von Zeichenketten mit „wildcard"-Zeichen 15
2.3.3 Vergleichen von Zeichenketten („Wortsymbole") 17
2.3.4 Erkennen von Wortklassen 19
2.4 Musterumsetzung 24
2.4.1 Ersetzen von Zeichenketten 24
2.4.2 Bearbeiten numerischer Zeichenketten 25
2.4.2.1 Nichtnumerische Zeichen durch „0" ersetzen 25
2.4.2.2 Zeichenketten in Integer umwandeln 26
2.4.2.3 Umwandlung von Zeichenketten in Bytes 27
2.4.2.4 Umwandlung von Zahlen in Zeichenketten 28
2.4.3 Formataufbereitung von Zeichenketten 29
2.4.3.1 Ausgabe von Texten linksbündig 29
2.4.3.2 Ausgabe von Texten rechtsbündig 30
2.4.4 Matrizenroutinen 31
2.4.4.1 Initialisierung von Matrizen 32
2.4.4.2 Kopieren von Matrizen 33
2.4.4.3 Spur einer Matrix 34
2.4.4.4 Addition und Subtraktion von Matrizen 35
2.4.4.5 Multiplizieren von Matrizen 36
2.4.4.6 Inversion von Matrizen 38
2.4.5 Zeichenübersetzung 40
2.4.5.1 Umwandlung von Groß- in Kleinbuchstaben 40
2.4.5.2 Umwandlung von ASCII- in EBCDI-Zeichen 41
2.5 Datensicherung (Texte codieren und decodieren) 43

3 Standardisierung der Bildschirmein- und -ausgabe 45

3.1 Inkompatibilität verschiedener MS-DOS-Rechner 45
3.2 Erhöhung der Portabilität von Programmen durch standardisierte Ein- und Ausgaberoutinen 46
3.2.1 Standardisierte Tastaturabfrage und Interpretationsroutine 47
3.2.2 Standardisierte Attributdefinitionen (z.B. Invers, Blinken, Unterstreichen) 51
3.2.3 Bildschirminitialisierung 53
3.2.4 Standardbildschirmausgabe 53
3.2.5 Formatierte Bildschirmeingabe- und -editierfunktion 54
3.2.6 Optische Unterstützungsfunktion bei der Bildschirmausgabe 58

4 Interface Turbo Pascal und MS-DOS 62

4.1 Beschreibung des Interfaces 64
4.2 Ausgewähltes, ausführliches Beispiel: PURGE-Utility 66

5 Bitmap-Techniken 73

5.1 Bitposition umrechnen 74
5.2 Bitmaps initialisieren 75
5.3 Bit setzen bzw. Bit zurücksetzen 76
5.4 Bit testen 76
5.5 Nächstes freie Bit suchen 77
5.6 Speicherplatz komprimieren 78

6 Bearbeitung von Datums- und Zeitfunktionen 84

6.1 Verifizierung von Datumseingaben 84
6.2 Datumsberechnungen 85
6.2.1 Berechnung der Anzahl Tage seit 1900 85
6.2.2 Errechnen von Terminspannen 86
6.2.3 Ermittlung des Wochentags 87
6.2.4 Prüfung des Jahres auf 53 Kalenderwochen 87
6.3 Zeitberechnungen 88
6.3.1 Zeitangaben prüfen 88
6.3.2 Berechnung von Zeitspannen 89

Anhang 91

Literaturhinweise 91
Anhang A1 Glossar der verwendeten Begriffe 92
Anhang A2 Liste der (dokumentierten) Funktionen in MS-DOS 97

Lösung der Übungsaufgaben 122

Sachwortverzeichnis 147