Problemlösen mit Oberon

(Oberon−System Version 4 − 2.3)

von Michael Fothe

Stand: September 2002

Im Informatik−Unterricht an allgemein bildenden Schulen erwerben die Schülerinnen und Schüler informatische Bildung. Studium oder Berufsausbildung sollen vorbereitet, jedoch nicht vorweggenommen werden. Ein wichtiges Thema ist nach wie vor das Bearbeiten von Problemen mit einer höheren Programmiersprache. Der Autor setzt im eigenen Unterricht seit 1995 die Programmiersprache Oberon ein (siehe Anhang 1).

Ziel dieses Beitrags ist das Vorstellen der CD−ROM "Problemlösen mit Oberon". (Alle Kapitel−Angaben beziehen sich auf die CD−ROM Nr. 59.) Mit der CD−ROM wird ein Schritt in Richtung eines interaktiven elek−tronischen Lehrbuchs gegangen. Als Oberon−System wird die Version 4 −2.3 für Windows−Computer verwendet (siehe Anhang 2). Die Version 4 − 2.3 gibt es für weitere Plattformen (z. B. Linux). Erfahrungen zeigen, dass es den Schülerinnen und Schülern mit akzeptablem Aufwand gelingt, sich in die integrierte Oberon−Arbeitsumgebung einzuarbeiten. Sie erschließen sich eine "Welt" mit vielen Möglichkeiten.

In diesem Beitrag wird dargestellt,

- was man unter einem Oberon−Programm versteht,
- in welchen Schritten ein Oberon−Quelltext erarbeitet wird,
- wie sich die Informationen textuell zusammenfassen lassen, die zur Lösung einer Aufgabe gehören, und
- welche didaktisch−methodischen Möglichkeiten sich aus den verschiedenen Arten der Ein− und Ausgabe ergeben.

Es werden Oberon−Programme vorgestellt, die experimentelles Arbeiten und arbeitsteiliges Problemlösen unterstützen sollen. Es wird deutlich werden, dass die Übergänge zwischen Programmierer und Nutzer im Oberon−System fließend sind. Die Schülerinnen und Schüler sollen erfahren, wie mit Hilfe von Informatik−Werkzeugen Komplexität bewältigt werden kann. Dem modularen Programmieren kommt dabei eine Schlüsselrolle zu.

CD−ROM "Problemlösen mit Oberon"

Die CD−ROM besteht aus 18 Kapiteln. Sie enthält Oberon−Quelltexte und zahlreiche Texte mit Entwürfen, Erläuterungen, Beispielen und Aufgaben. Die Inhalte beziehen sich auf das Grundfach Informatik an den Thüringer Gymnasien. Im Mittelpunkt stehen daher strukturiertes und modulares Programmieren. Im Unterricht des Autors besitzt das objektorientierte Programmieren in Projektarbeiten eine wichtige Rolle. Nach einer Einführung in die Grundprinzipien des objektorientierten Programmierens arbeiten sich interessierte Schüler mit Hilfe der zur Verfügung stehenden Literatur recht umfassend in die OOP−Technik ein (vgl. Hermes & Stein (1996) und Mössenböck (1998)).

Problemlösen mit Oberon

Oberon−Programm

Ein Oberon−Programm besteht aus einem Tool und einem oder mehreren Modul−Quelltexten. Ein Tool enthält Kommandos und Erläuterungen. Ein Quelltext ist ein nach der Oberon−Syntax strukturierter Text. In Oberon gibt es kein Hauptprogramm, das als Ganzes aufgerufen wird.

Erarbeiten eines Quelltextes

Das Erarbeiten eines Oberon−Quelltextes soll an dem folgenden Spiel erläutert werden (Kapitel 5): Der Computer erzeugt eine zufällige Folge von vier Ziffern. Der Mensch soll die Folge ermitteln. Dazu gibt er eine Folge von vier Ziffern in den Computer ein, die vom Computer bewertet wird.

Aus der Aufgabenstellung ergibt sich, dass zwei nutzerspezifische Kommandos benötigt werden. Ein Kommando K_1 zum Erzeugen einer zufälligen Folge von vier Ziffern und ein Kommando K_2 zum Bewerten einer Folge von vier Ziffern. Ein Kommando steht für einen Prozess. Das Kommando K_1 hat keine Eingaben. Das Kommando K_2 hat als Eingabe vier ganze Zahlen.

Zu jedem Kommando gehört eine exportierte parameterlose Prozedur. Solche Prozeduren nennen wir Kommandoprozeduren. Zu den Kommandos K_1 und K_2 sollen die Kommandoprozeduren **Erzeugen** und **Bewerten** des Moduls **Ziffern** gehören. Mit **Edit.Open Ziffern.Mod~**

wird ein leeres Textfenster angelegt, in das der Quelltext des Moduls **Ziffern** geschrieben wird.

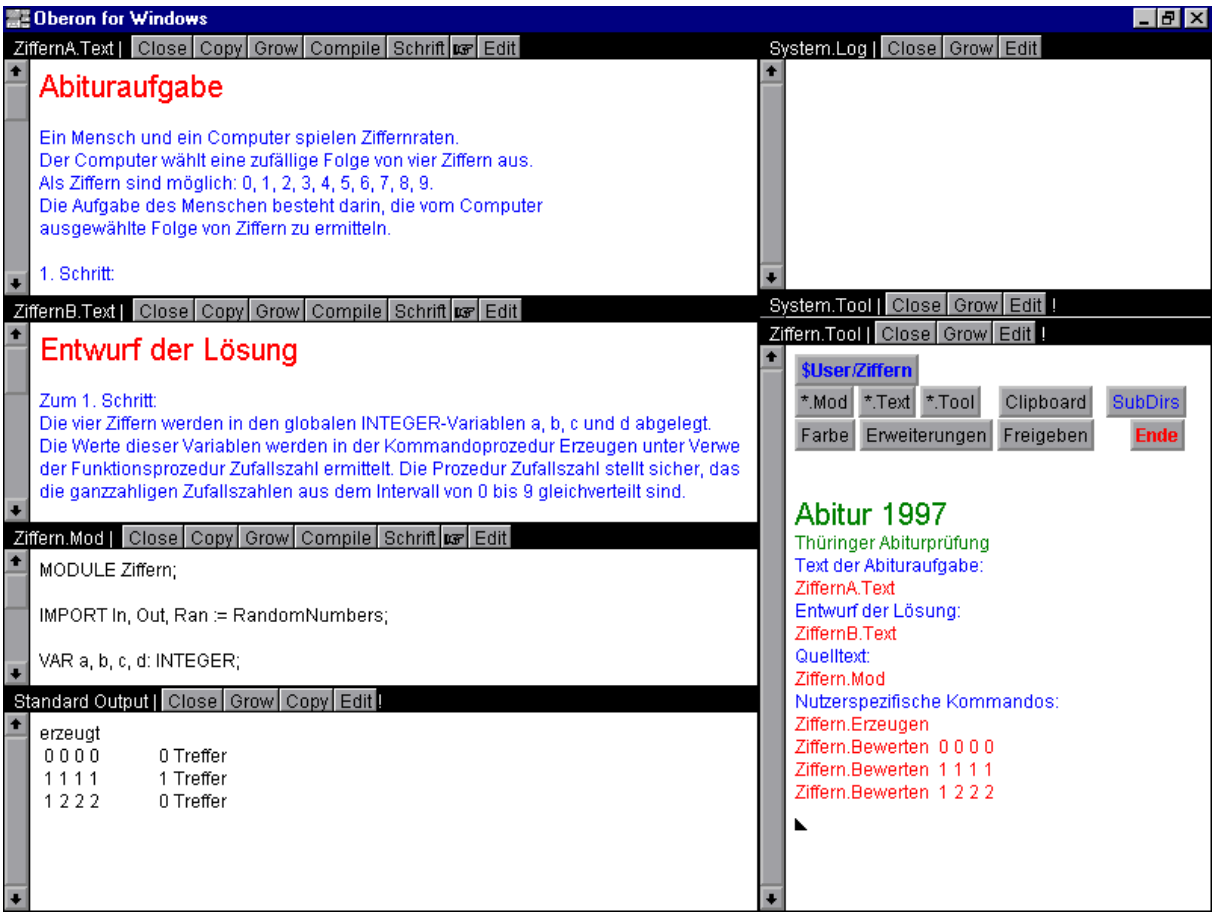

Abbildung 1

Zu den Variablen werden die folgenden Entscheidungen getroffen: Das Modul **Ziffern** erhält die vier globalen INTEGER−Variablen **a**, **b**, **c** und **d**. Diesen Variablen werden in der Kommandoprozedur **Erzeugen** zufällige Werte zugewiesen. Globale Variablen eines Moduls behalten ihren Wert dauerhaft. Die Kommandoprozedur **Bewerten** erhält die fünf lokalen INTEGER−Variablen **e**, **f**, **g**, **h** und **anzahl**. In den Variablen wird die eingelesene Folge von Ziffern bzw. die Anzahl an Treffern gespeichert. Außer den beiden Kommandoprozeduren ist eine private (nicht−öffentliche) Funktionsprozedur **Zufallszahl** zu realisieren. Diese Funktionsprozedur wird in der Kommandoprozedur **Erzeugen** aufgerufen. Nach dem erfolgreichen Übersetzen des Quelltextes des Moduls **Ziffern** stehen die Kommandos K_1 und K_2 zur Verfügung des Nutzers. Die beiden nutzerspezifischen Kommandos besitzen mit **Ziffern.Erzeugen** und **Ziffern.Bewerten** qualifizierte Namen. Die Kommandos werden in das Tool **Ziffern.Tool** geschrieben (siehe Abbildung 1). Ein Kommando wird ausgeführt, indem sein Name mit der

mittleren Maustaste (oder ersatzweise mit der Strg−Taste der Tastatur) angeklickt wird. Mit **Out.Open** wird das Ausgabefenster geöffnet.

Erarbeiten eines Tools

In einem Tool werden alle wichtigen Informationen zusammengefasst, die zur Lösung einer Aufgabe gehören.

- Ein Tool ist ein Textfenster.
- Ein Textfenster ist aus Textelementen aufgebaut (z.B. aus Zeichen, Buttons, Faltungen und Animationen).
- Jedes Textfenster besitzt einen Editor.
- Das Kopieren und Löschen von Textteilen geschieht mit Mausoperationen oder dem Clipboard.

Zum Beispiel besteht das Tool für die polyalphabetische Substitution aus den folgenden Bestandteilen (Kapitel 10):

Namen der Textfenster	Quelltext des Moduls Poly: **Poly.Mod** Erläuterungstext: **Poly.Text** Klartext, der zu verschlüsseln ist: **Klar.Text** Geheimtext, der zu entschlüsseln ist: **Geheim.Text**
System−Kommandos	Buttons, die stellvertretend für System−Kommandos stehen
Nutzer−spezifische Kommandos	Kommando zum Einlesen eines Schlüssels: **Poly.Schluesseleingabe** Kommando zum Verschlüsseln eines Klartextes: **Poly.Verschluesseln** Kommando zum Entschlüsseln eines Geheimtextes: **Poly.Entschluesseln**
Erläuterungen	Schrittfolgen, die vom Nutzer zur Schlüsseleingabe, zum Verschlüsseln des Klartextes und zum Ent−schlüsseln des Geheimtextes auszuführen sind

Vielfalt bei den Eingaben

Für Eingaben gibt es drei Varianten:

A	Eingaben können nach dem Namen eines Kommandos stehen.
B	Ein Stern nach dem Namen eines Kommandos bedeutet, dass die Eingaben aus dem Textfenster eingelesen werden, welches mit Hilfe der F1−Taste mit einem Stern markiert wurde.
C	Ein Pfeil nach dem Namen eines Kommandos bedeutet, dass von dem Zeichen an eingelesen wird, welches invers unter Verwendung der rechten Maustaste markiert wurde.

1. Beispiel:

Variante A wurde bereits bei dem Kommando **Ziffern.Bewerten** verwendet. Mit einer Tilde (~) kann der Eingabestrom explizit abgeschlossen werden.

2. Beispiel:

Beim Suchen in einem Labyrinth (Kapitel 12) gibt es das Labyrinth dreifach (extern, intern, grafisch). Die externe Darstellung des Labyrinths in einem Textfenster ermöglicht ein leichtes Verändern des Labyrinths durch den Nutzer (siehe Abbildung 2) Der Text besteht aus den Zeichen 0, 1, 2, 3 und 4. Nullen markieren die Wege, die bei der Suche betreten werden dürfen. Einsen stehen für Mauern oder Hecken, die nicht zu betreten sind. Zweien fassen das rechteckige Labyrinth ein. Eine Drei markiert das Startfeld, eine Vier das Zielfeld. Das Labyrinth wird mit Variante B eingelesen. Die interne Darstellung als zweidimensionale Reihung ist Voraussetzung für die schnelle Suche von zulässigen Wegen vom Start− zum Zielfeld durch einen Backtracking−Algorithmus. Das Labyrinth wird zusätzlich im Grafikfenster dargestellt. Der Nutzer kann dadurch die Suche nach den zulässigen Wegen verfolgen ("Faden der Ariadne").

3. Beispiel:

Bei der monoalphabetischen Substitution (Kapitel 9) gibt es eine feste Zuordnung der Klar− zu den Geheimbuchstaben, die Sender und Empfänger einer Nachricht vereinbaren. Die Zuordnung lässt sich als permutiertes Alphabet angeben. Ein Beispiel ist KWAMUGHQFTI CEZLDPRYBJVONSX. (Der Klarbuchstabe a wird dem Geheimbuchstaben K zugeordnet. b wird W, c wird A, d wird M zugeordnet usw.) Im Textfenster **Alphabete.Text** stehen mehrere

permutierte Alphabete zur Auswahl. Das Alphabet, das genommen werden soll, wird mit Variante C vom Kommando **Mono.ErstesAlphabet** eingelesen. Klar− und Geheimtext befinden sich in den Textfenstern **Klar.Text** bzw. **Geheim.Text**. Diese Texte werden von den Kommandos **Mono.Verschluesseln** bzw. **Mono.Entschluesseln** mit Variante B eingelesen.

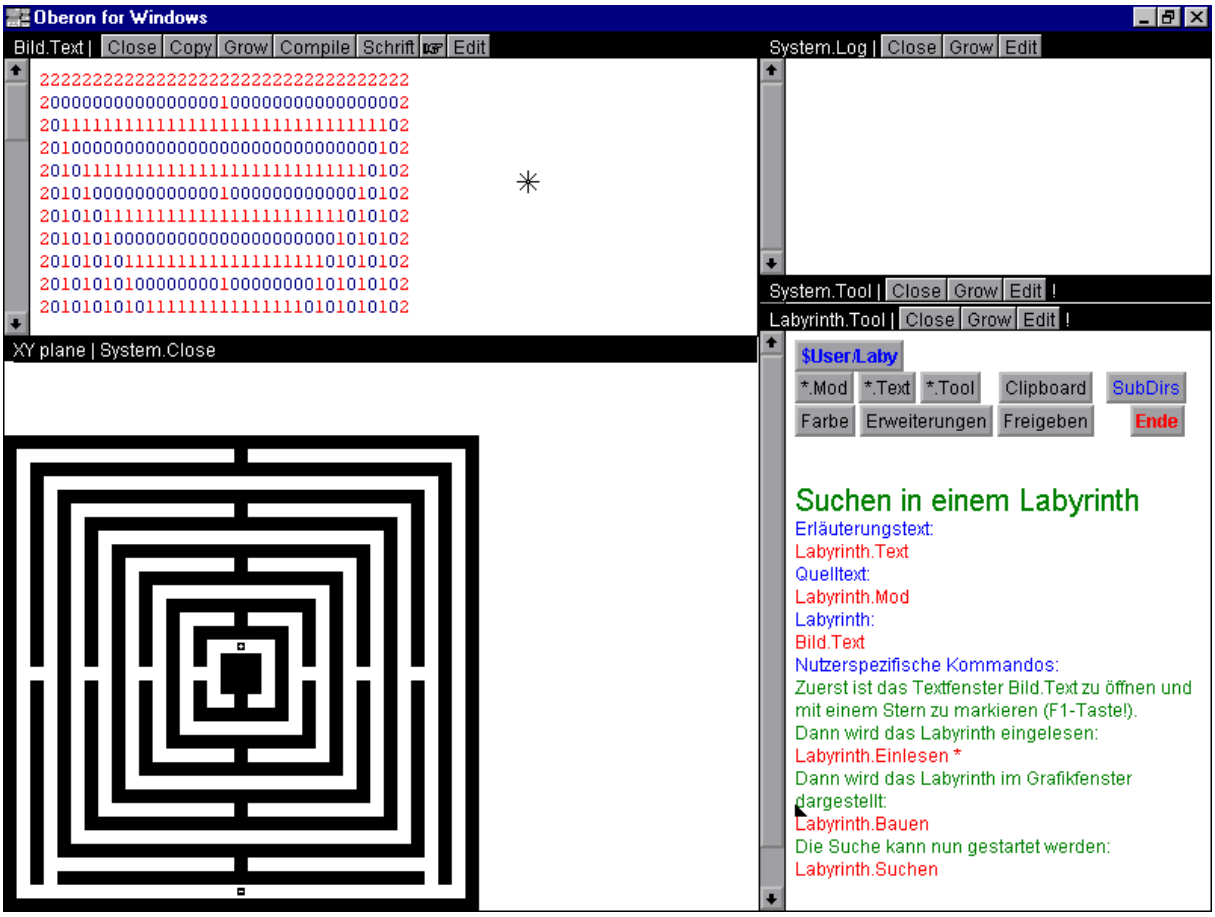

Abbildung 2

4. Beispiel:

Die Ergebnisse der Erfassung des stündlichen Verkehrsaufkommens an drei Kreuzungen sollen von einem Oberon−Programm ausgewertet werden (Kapitel 6). Jeweils 24 INTEGER−Zahlen werden in ein Text−fenster geschrieben (z. B. **Kreuzung1.Text**). Es ist nicht erforderlich, die Zahlen intern in einer Reihung zu verwalten. Vielmehr werden die Zahlen immer wieder neu mit Variante B aus dem entsprechenden Text−fenster eingelesen.

Vielfalt bei den Ausgaben

Die Ausgaben eines Oberon−Programms werden in das Ausgabefenster geschrieben und können

- kommentiert,
- ausgedruckt,
- gespeichert oder
- wieder zu Eingaben werden.

Die Möglichkeiten werden an den Oberon−Programmen "Quicksort" und "Hilbert−Kurven" verdeutlicht (Kapitel 15 und 16). Damit wird gezeigt, wie sich experimentelles Arbeiten und arbeitsteiliges Problemlösen im Oberon−System realisieren lassen.

1. Beispiel:

Für die Zeitmessung bei Quicksort gibt es die beiden nutzerspezifischen Kommandos **QuicksortZ.Art** und **QuicksortZ.Sortieren**. Für das Kommando **QuicksortZ.Art** sind zwei Eingaben bereitzustellen. Die erste Eingabe ist eine 1, 2 oder 3. Sie gibt an, welche Eigenschaft die zu sortierenden Zahlen besitzen. Die zweite Eingabe ist eine 1, 2, 3 oder 4. Sie gibt an, welches Element als Trennelement genommen wird. Sollen z. B. Zufallszahlen sortiert werden und soll als Trennelement ein zufälliges Element einer jeden Teilfolge genommen werden, so ist **QuicksortZ.Art 2 4~** auszuführen. Danach kann zum Beispiel **QuicksortZ.Sortieren 10000~** zum Sortieren von 10000 REAL−Zahlen ausgeführt werden. Die benötigte Rechenzeit wird vom Programm ermittelt (z. B. 41 ms) und in das Ausgabefenster wird **2 4 10000 41** geschrieben. In einem ersten Experiment werden für eine feste Anzahl von zu sortierenden Zahlen (z. B. 10000) die zwölf Fälle bearbeitet. Die Messergebnisse werden im Ausgabefenster durch Ordnen und Kommentieren weiter bearbeitet. Dadurch entsteht ein Protokoll, das ausgedruckt oder gespeichert werden kann. In einem zweiten Experiment wird die Anzahl zu sortierender Zahlen bei fester Eigenschaft und Wahl des Trennelementes variiert. Es können bis zu 200000 REAL−Zahlen sortiert werden. Die Messergebnisse können mit den Kommandos **GrafikQ.Skalieren** und **GrafikQ.Zeichnen** aus dem Ausgabefenster eingelesen und im Grafikfenster in Diagramm−Form dargestellt werden.

2. Beispiel:

Beim Erarbeiten eines Oberon−Programms zum Zeichnen von Hilbert−Kurven können zwei Schülerinnen oder Schüler gleichzeitig arbeiten. Das Kommando, das von dem einen Schüler geschaffen wird, liest die Ordnung der Hilbert−Kurve ein und gibt eine Folge von Zeichenanweisungen aus. Für die Hilbert−Kurve der Ordnung 2 wird SWNWWS ESWSEENES ausgegeben. Das Kommando, das von dem anderen Schüler geschaffen wird, liest eine Folge von Zeichenanweisungen ein und erstellt die Zeichnung. Beide Schüler können gleichzeitig arbeiten und die realisierten Kommandos unabhängig voneinander testen. Die Kommunikation erfolgt transparent über ein Textfenster.

Themen für den Informatik−Unterricht

1. Thema: Sortieren

Die Schülerinnen und Schüler sollen mit sechs Oberon−Programmen beim Erlernen des Sortier−Algorithmus "Sortieren durch Auswählen" und beim Ermitteln des Zeitverhaltens des Algorithmus unterstützt werden (Kapitel 14).

Eine Übersicht zu den Programmen:

1	Es werden INTEGER−Zahlen sortiert, die in einem Textfenster gegeben sind. Die Schülerinnen und Schüler lernen den Algorithmus "Sortieren durch Auswählen" kennen.
2	Datensätze werden nach ihrer zeitlichen Reihenfolge sortiert. Als Schlüssel wird ein Kalenderdatum genutzt. Es wird das Modul **DatumOperationen** importiert und die logische Funktionsprozedur **LiegtVor** wird zum Vergleich zweier Schlüssel verwendet.
3	Es werden Strecken sortiert, die auf einer Geraden liegen und von denen die ganzzahligen Koordinaten des Anfangs− und Endpunktes bekannt sind. Als Schlüssel wird die Koordinate des Anfangspunktes genutzt.
4	Der Algorithmus wird visualisiert. Dazu wird eine Punktwolke sortiert. Jeder Sortier−Algorithmus, so auch Sortieren durch Auswählen, liefert eine spezifische Animation.
5	Die Rechenzeiten werden gemessen und grafisch dargestellt.

6	Die Operationen werden gezählt. Aus einem Textfenster werden INTEGER−Zahlen eingelesen (10, 20 oder 50 Zahlen mit verschiedenen Eigenschaften: unsortiert, aufsteigend sortiert, absteigend sortiert). Die Zahlen werden sortiert. Für jede Vergleichs− bzw. jede Bewegungsoperation wird ein Stern im Ausgabefenster ausgegeben. Jedes Mal, wenn eine Zahl an die richtige Position gebracht wurde, erfolgt ein Zeilenwechsel. Die speziell auf das Sortieren durch Auswählen abgestimmte Methode soll die Schülerinnen und Schüler beim Erforschen der Ursachen für das Zeitverhalten des Algorithmus unterstützen.

2. Thema: Kalenderdaten

Kalenderdaten sind Objekte, die im Alltag der Schülerinnen und Schüler vorkommen. Eine Diskussion zum Jahr−2000−Problem kann den Ausgangspunkt für die Behandlung des Themas "Kalenderdaten" im Informatik−Unterricht darstellen.

Im Oberon−System lassen sich "Lehrprogramm−Modus" und "Program−miermodus" realisieren (vgl. Fothe (1984)). Dazu wird eine wesentliche Eigenschaft von Kommandos ausgenutzt: Sowohl System−Kommandos als auch nutzerspezifische Kommandos sind Bestandteile von Texten und werden aus dem Text heraus ausgeführt. Im Kapitel 3 befindet sich der Text **Datum.Text**. Er enthält Erläuterungen zum Gregorianischen Kalender und Aufgaben, die von den Schülerinnen und Schülern gelöst werden sollen. Der Text entspricht dem Lehrprogramm−Modus. Aus diesem Modus kann mit Hilfe des System−Kommandos **System.Open** in den Programmiermodus gewechselt werden. Es wird ein vorgegebenes Tool geöffnet, die Schülerinnen und Schüler lösen eine Aufgabe und kehren anschließend in den Lehrprogramm−Modus zurück. Dazu werden von ihnen die bearbeiteten Texte gespeichert und die Textfenster geschlossen. Typische Aufgaben sind das Analysieren und das Modifizieren eines vorgegebenen Oberon−Programms. Zum Beispiel wird mit **System.Open Fuenf.Tool~** das Tool **Fuenf.Tool** geöffnet. Mit **Edit.Open Daten.Mod~** wird dann der Oberon−Quelltext **Daten.Mod** geöffnet. Die Schülerinnen und Schüler haben die Aufgabe, die im Modul **Daten** vorhandene Kommandoprozedur **Naechster** (zum Ermitteln das Datums von morgen) zu analysieren und die Kommandoprozedur **Vorhergehender** (zum Ermitteln des Datums von gestern) zu entwerfen und zu implementieren. Beispiele zum Testen der Kommandoprozedur **Vorhergehender** sind in dem Tool vorgegeben (siehe Abbildung 3).
Im Kapitel 17 wird ein Modul **DatumOperationen** mit dem Datentyp **Datum** (einem Verbund) und zwölf Operationen für Kalenderdaten

bereitgestellt. Das Modul **DatumOperationen** wird z. B. vom Modul **Kalender** importiert und beim Ermitteln eines Jahreskalenders für ein eingelesenes Jahr genutzt. Das Modul **DatumOperationen** besitzt eine Prozedurschnittstelle, das Modul **Kalender** besitzt eine Kommandoschnittstelle. Die Schülerinnen und Schüler sollen erkennen, dass modulares Programmieren eine Möglichkeit ist, Komplexität zu bewältigen.

Abbildung 3

3. Thema: Abstrakte Datentypen

Für die Arbeit mit einfach verketteten Listen wurde ein abstrakter Datentyp Liste spezifiziert und als Modul **adtliste** realisiert (Kapitel 18). Mehrere Module importieren und nutzen das Modul **adtliste**: das Modul **eins** (einfaches Beispiel), das Modul **zwei** (Verwalten zweier Stapel) und das Modul **drei** (Anlegen einer sortierten Liste). Die Schnittstelle des Moduls **adtliste** wird mit den System−Kommandos **Browser.ShowDef** oder **Def.Show** aus dem Quelltext des Moduls gewonnen. Es gibt von außen keinen Hinweis auf die Art der Imple−mentierung der einfach verketteten Listen. Die Schülerinnen und Schüler

sollen erkennen, dass abstrakte Datentypen eine Weiterführung des Geheimnisprinzips sind und dass sich mehrere Variablen des abstrakten Datentyps Liste deklarieren lassen.

Resümee

Nach den vorliegenden Erfahrungen ist die Programmiersprache Oberon ein Werkzeug zum Problemlösen, das sich für den Einsatz im Informatik−Unterricht an allgemein bildenden Schulen eignet. Die CD−ROM "Problemlösen mit Oberon" kann die unterrichtenden Lehrerinnen und Lehrer beim Bereitstellen eigenen elektronischen Materials für den Einsatz in ihrem Unterricht unterstützen. Auf den Kenntnissen und Fähigkeiten, die die Schülerinnen und Schüler im Informatik−Unterricht an den allgemein bildenden Schulen mit Oberon erworben haben, kann in Studium oder Berufsausbildung beim Einarbeiten in Java, C++ oder andere Programmiersprachen aufgebaut werden. Auf aktuelle Entwicklungen (komponentenbasiertes Programmieren mit dem ETH−Oberon, Active Oberon) kann an dieser Stelle nur hingewiesen werden.

Literatur

Fothe, Michael: Zur Computersimulation im Chemieunterricht. In: 2. Seminar "Computer als Mittel und Gegenstand der Ausbildung" Ottendorf−Okrilla 15.−19. Oktober 1984. Dresdner Reihe zur Forschung 13/84, ISSN 0233−0164

Fothe, Michael: Problemlösen mit Oberon. Nr. 38 der Reihe "Materialien" des Thüringer Instituts für Lehrerfortbildung, Lehrplanentwicklung und Medien (ThILLM), Bad Berka 2000 (CD−ROM), ISSN 0944−8691

Fothe, Michael; Kämmerer, Manfred; Lotze, Annemarie: Strukturiertes und modulares Programmieren mit Oberon. Nr. 59 der Reihe "Materialien" des ThILLM, Bad Berka 2001 (CD−ROM)

Fothe, Michael: Problemlösen mit Oberon. Konzeption und Einsatz eines elektronischen Lehrbuchs. LOG IN 21 (2001) Heft 2, S. 33−40 *(Grundlage dieses Textes)*

Fothe, Michael: Rechnen auf den Linien. Vier Spiele mit dem Computer. LOG IN 21 (2001) Heft 3/4, S. 48−53

Goedicke, Michael: Java in der Programmierausbildung: Konzept und erste Erfahrungen. Informatik−Spektrum 20 (1997) 6, S. 357−363

Hermes, Alfred: OOP im Unterricht. Ein Plädoyer für einen gleitenden Paradig−menwechsel (Teile 1 und 2). LOG IN 16 (1996) 4, S. 29−33 und LOG IN 16 (1996) 5/6, S. 62−67

Hermes, Alfred; Stein, Clemens: Objektorientierte Programmierung. Ein Zugang in Oberon mit Anwendungen von Listen und Grafik. Ernst Klett Verlag Stuttgart 1996

Hug, Karlheinz; Ketz, Helmut: Objektorientierung mit Oberon−2 in der Ingenieur−Grundausbildung. Informatik−Spektrum 20 (1997) 6, S. 350−356

Marincek, B.; Marais, J. L.; Zeller, E.: OBERON. Ein Kurzleitfaden für Studenten. Vieweg Braunschweig, Wiesbaden 1999

Mössenböck, Hanspeter: Objektorientierte Programmierung in Oberon−2. 3. Auflage. Springer−Verlag Berlin, Heidelberg, New York 1998

Nikitin, Eric: Into the Realm of Oberon. Springer New York, Berlin, Heidelberg 1998

Reiser, Martin; Wirth, Niklaus: Programmieren in Oberon. Das neue Pascal. Addison−Wesley Bonn, Paris, Reading 1997

von Lavergne, Harro: Thesen zur aktuellen Orientierungslosigkeit. LOG IN 18 (1998) 5, S. 19−23

Wirth, N.: Gedanken zur Software−Explosion. Informatik−Spektrum 17 (1994) 1, S. 5−10

The School of Niklaus Wirth: The Art of Simplicity. Herausgegeben von Böszörmenyi, Laszlo; Gutknecht, Jürg; Pomberger, Gustav. dpunkt Verlag 2000

Anhang 1

Programmiersprache Oberon

Oberon ist eine kleine Sprache, die in der Tradition von Algol 60, Pascal und Modula−2 steht. Oberon unterstützt strukturiertes, modulares und objektorientiertes Programmieren. Die von Turbo Pascal bekannten Kontrollstrukturen, Datentypen, Parameterarten bei Prozeduren und anderes finden sich meist in Oberon wieder. Aufzählungs− und Teilbe−reichstypen gibt es nicht; Groß− und Kleinbuchstaben werden unterschieden (vgl. Reiser & Wirth (1997), Marincek, Marais & Zeller (1999) und Nikitin (1998)).

In Oberon ist das Modul von zentraler Bedeutung:

- Ein Oberon−Modul wird vom Oberon−Compiler in ausführbaren Maschinencode übersetzt (Objekt−Modul).
- Ein Stern hinter dem Namen einer Prozedur, eines Typs, einer Variablen oder einer Konstanten markiert diese Prozedur, diesen Typ, diese Variable oder diese Konstante als exportiert (öffentlich). Sie können also von einem anderen Modul importiert werden. Der Stern heißt Exportmarkierung.
- Die Module, die von einem Modul importiert werden sollen, sind in die Import−liste aufzunehmen.
- Die importierten Prozeduren, Typen, Variablen und Konstanten werden stets mit einem qualifizierten Namen angesprochen. Zum Beispiel steht der qualifizierte Name **DatumOperationen.Datum** für den Typ **Datum** aus dem importierten Modul **DatumOperationen**.

Die Möglichkeiten des objektorientierten Programmierens mit Oberon werden mitunter kritisch bewertet. Der Wert von Oberon als Ausbildungssprache wird jedoch hervorhoben (vgl. Goedicke (1997) und Hug & Ketz (1997)). Eine didaktisch−methodische Konzeption zu OOP mit Oberon liegt vor (vgl. Hermes (1996) und von Lavergne (1998)).

Anhang 2

Oberon−System Version 4

Wesentliche Kennzeichen des Oberon−Systems Version 4 sind:

- Kommandos, die spezielle Dialogformen zur Folge haben,
- Textfenster, die sich nicht vollständig verdecken (falls das System−Kommando **System.Grow** nicht ausgeführt wurde),
- dynamisches Laden von Modulen,
- automatische Speicherbereinigung,
- Typinformationen zur Laufzeit,
- die Möglichkeit, das Oberon−System den persönlichen Vorstellungen anzupassen,
- die variantenreiche Nutzung einer Dreitastenmaus,
- die geringe Verwendung von Warnungen und
- die Tatsache, dass Oberon−Programme stets im Oberon−System ausgeführt werden.

Unter www.th.schule.de/th/lfk−informatik ist eine Schüler−Version der CD−ROM kostenlos erhältlich − und zwar für die Oberon−Systeme Version 4 − 1.41, Version 4 − 2.3 und Pow! . Nach dem Entpacken und Kopieren des Oberon−Systems Version 4 − 2.3 können Sie nähere Informationen auf folgendem Weg erhalten: Mit Hilfe des Buttons SubDirs (anklicken mit der mittleren Maustaste oder ersatzweise mit der Strg−Taste der Tastatur) wird in die Unterverzeichnisse User und dann Einstieg gewechselt. Anschließend Öffnen von Einstieg.Tool mit dem Button *.Tool. Dort befinden sich vier wichtige Texte zur Arbeit mit dem Oberon−System.