Die Programmiersprache Oberon

Von Michael Nejez, [myk3@csorg.at], Nov. 2004
Bruckhaufner Haupstr. 15, A-1210 Wien

Kurzzusammenfassung:

Oberon ist eine hoch entwickelte Programmiersprache, die auf vielen Betriebssystemen verwendet werden kann, beziehungsweise sogar als eigenes Betriebssystem fungiert. Die Dokumentation ist absichtlich kurz gehalten und besteht aus rund 25 Seiten, die sämtliche Konzepte der Sprache im wesentlichen beschreibt, was Oberon zu einer verhältnismäßig leicht erlernbaren Sprache macht. Hinzu kommen zahlreiche Parallelen zu Modula-2 und Pascal, sowie zu Java und anderen Sprachen, sodass man sich vor allem dann relativ schnell in die Sprache einfinden kann, wenn man schon gewisse Programmier-Vorkenntnisse hat. Mit Oberon lassen sich sowohl kleine als auch größere Projekte realisieren, die sonst klassischen Programmiersprachen vorbehalten waren, wobei allerdings mit Oberon nicht so schnell laufende Endresultate erzielt werden können wie mit den klassischen Sprachen, dafür ist die Entwicklungszeit in Oberon bei vielen Anwendungsbereichen als kürzer einzustufen.

Einführung:

Oberon ist eine objektorientierte Programmiersprache die von Niklaus Wirth und Jürg Gutknecht in Zusammenarbeit mit der ETH Zürich entwickelt wurde. Niklaus Wirth gilt auch als Entwickler der Programmiersprache Pascal, was gewisse Ähnlichkeiten zwischen den beiden Sprachen erklärt. Oberon ist universell einsetzbar, so existiert z.B. das ETH Oberon System als eigenständiges Betriebssystem der ETH Zürich, das in der Sprache Oberon implementiert ist und als Entwicklungsgrundlage für die Sprache diente. Des weiteren ist es auch ähnlich zu Modula-2 als dessen Weiterentwicklung es gilt und von dem es sich vor allem durch das Konzept der Typerweiterung unterscheidet, was es ermöglicht, neue Datentypen auf der Basis bereits Existierender zu definieren und Relationen zwischen den beiden herzustellen.

Syntax und Semantik:

Ein Kommando ist die elementare Ausführungseinheit für das Oberon-System. Kommandos sind gebündelt in Modulen. Datenhaltung und konsistente Datenübergabe wird jeweils per Modul definiert. Ein Modul kann ausser extern zugänglichen Kommandos sowohl intern als auch extern ausführbahre Prozeduren enthalten. Sichtbarkeit und Zugriffsbedingungen für Daten und Prozeduren festzulegen und zu überwachen sind Aufgaben der Module.

Alle Komponenten von Oberon sind erlaubte Quellen für Kommandos. Es spielt keine Rolle, wo das Kommando eingegeben wird. Dies gibt beträchtliche Flexibilität, aber es ist auch eine denkbare Fehlerquelle. Anstelle eines Kommandos kann versehentlich irgendein Text aktiviert werden. Oberon muss in der Lage sein, zumindest grobe Fehler und Probleme abzufangen. Dies kann anhand von zwei verschiedenen Vorgehensweisen geschehen:

1. Syntaktische Analyse.
Es gibt eine eindeutig festgelegte Syntax, der alle Kommandos folgen. Wenn eine Eingabe nicht dieser Syntax folgt, dann kann sie kein Kommando sein.

2. Semantische Analyse.
Wenn eine Eingabe syntaktisch akzeptabel ist, so legt die Syntax für die einzelnen Bestandteile bestimmte Rollen nahe. In einem zweiten Schritt kann Oberon prüfen, ob die Bestandteile eine Interpretation haben, die dieser Rolle gerecht werden kann. Um dies zu klären versucht Oberon, die Eingabe semantisch zu analysieren, das heisst die Bedeutung zu identifizieren.

Module, Prozeduren und Kommandos werden in Oberon mit den Zeichen von a-z, A-Z und 0-9 benannt, wobei das erste Zeichen des Namens ein Buchstabe sein muss.

Ganze Zahlen werden in Oberon als Folge der Ziffern von 0-9 dargestellt, gefolgt von einem optionalen „H“ was die Zahl als Hexadezimalzahl identifiziert. Gleitkommazahlen enthalten einen Dezimalpunkt und können mit einer Potenz von 10 multipliziert werden, indem man an die Zahl ein „E“ oder ein „D“, sowie die gewünschte Zehnerpotenz angibt. (z.B. 3.4E5 = 3.4 mal 10 hoch 5)

Zeichenkonstanten können entweder durch das gewünschte Zeichen, eingeschlossen von Anführungszeichen definiert werden – oder alternativ durch die hexadezimale Ordnungszahl des Zeichens, gefolgt von einem X. Zeichenketten sind mehrere Zeichen, eingeschlossen von Anführungszeichen.

Zu erwähnen sind noch die Operatoren und Schlüsselwörter in Oberon, die ausschließlich aus Grossbuchstaben bestehen und nicht als Namen verwendbar sind.

Um die Lesbarkeit und Verständlichkeit von Programmen zu verbessern, gibt es natürlich auch in Oberon, wie in den meisten Programmiersprachen, die Möglichkeit, Kommentare in den Programmcode einzubinden. Diese müssen von Kommentarklammern (* *) eingeschlossen sein, was bewirkt, dass jeglicher Text der sich zwischen den Klammern befindet keinerlei Wirkung mehr auf das Programm hat.

Entscheidungen in Oberon:

Damit ein Programm Entscheidungen treffen kann, benötigt es gewisse Funktionen, die z.B. Variablen miteinander vergleichen können. Zu diesem Zweck gibt es in Oberon, wie auch in anderen Programmiersprachen sowohl die If- als auch die Case-

Anweisung. Der grundlegende Unterschied zwischen den beiden Anweisungen ist, dass bei der Case-Funktion genau eine Variable übergeben wird, deren Wert überprüft wird und darauf hin eine bestimmte, vordefinierte Operation durchgeführt wird. Bei der If-Anweisung hingegen wird überprüft ob gewisse Bedingungen erfüllt sind und anschließend wird die damit verknüpfte Operation ausgeführt. Für den Fall dass keine Bedingung zutreffend ist, besteht die Möglichkeit einen „else“-Block in die If-Abfrage oder auch in die Case-Anweisung zu integrieren, wodurch man eine Standardaktion definieren kann um Fehler zu umgehen. (z.B Fehlermeldung bei Eingabe nicht erlaubter Werte)

Schleifen:

For-Schleife:
Die Anweisungsfolge wird (Endwert-Anfangswert+1)-mal ausgeführt. Sie wird eingesetzt, wenn vor dem Eintritt in die Schleife bereits feststeht, wie oft die Wiederholung auszuführen ist.

Syntax:
FOR i:=Anfangswert TO Endwert DO
 <Anweisungsfolge>
END;

While-Schleife:
Die Anweisungsfolge wird so lange ausgeführt, wie die Eintrittsbedingung wahr bleibt.

Syntax:
WHILE Eintrittsbedingung DO
 <Anweisungsfolge>
END;

Repeat-Schleife:
Die Anweisungsfolge wird so lange ausgeführt bis die Austrittsbedingung wahr wird.

Daraus folgt, dass die Anweisungsfolge ohne besondere Maßnahmen stets mindestens einmal ausgeführt wird.

Syntax:
REPEAT
 <Anweisungsfolge>
UNTIL Austrittsbedingung;

Besonderheiten von Oberon:

Die wichtigsten Eigenschaften von Oberon sind strenge Datentyp-Prüfung, separate Compilation mit Typ-geprüften Schnittstellen und ein eindeutiger Namensraum (name-space). Des weiteren bietet es schreibgeschützen Export von Variablen, mehr-dimensionale, offene und dynamische Felder sowie Zeichenkettenoperationen. Oberon ist objekt-orientiert durch Datentyp-Erweiterung (Vererbung), Polymorphie, persistente Objekte, und Typ-gebundene Prozeduren (Methoden).

Vor- und Nachteile:

Oberon ist eine sehr hoch entwickelte Sprache, das heißt der programmiererfreundliche Quellcode unterscheidet sich sehr stark vom Maschinencode. Durch die großen Freiheiten die durch die modulare Programmstruktur zu Stande kommt und ein wesentliches Merkmal der Sprache ist, entstehen allerdings auch neue Fehlerquellen auf die besonders acht gegeben werden muss, wie zum Beispiel Gültigkeitsbereiche von Variablen. Mit Oberon können auch umfangreiche Anwendungen geschrieben werden, jedoch stellt sich die Frage nach der benötigten Geschwindigkeit, da diesbezüglich die klassischen Programmiersprachen wie C/C++ oder Java zumeist effizienter arbeiten. Dafür ist Oberon in Versionen für viele unterschiedliche Plattformen erhältlich (Windows, UNIX, Mac, Bluebottle) oder sogar als eigenes Betriebssystem (Native Oberon).

Literaturliste:

Der Großteil der Informationen wurde aus der Offiziellen Oberon – Dokumentation der ETH Zürich entnommen.

Niklaus Wirth: The programming language Oberon, Software—Practice & Experience 18 (7), (1988)

Andre Fischer, Hannes Marais: The Oberon Companion. A Guide to Using and Programming Oberon System 3, (1998)

Martin Reiser, Niklaus Wirth: Programmieren in Oberon. Das neue Pascal, Bonn (1994)

Niklaus Wirth, Jürg Gutknecht: Project Oberon. The Design of an Operating System and Compiler. Addison Wesley, (1992)