Abstrakte Datentypen – Dominik Rappaport, drbaden@ping.at

11. April 1999

Bevor wir uns mit abstrakten Datentypen beschäftigen, müssen wir zuerst auf die Datenabstraktion selbst genauer eingehen.

Datenabstraktion

Die Datenabstraktion unterscheidet sich von der Funktionsabstraktion dadurch, daß bei der letzteren nur der Algorithmus abstrahiert wird, bei der Datenabstraktion dagegen sowohl die interne Datenstruktur als auch ihre Zugriffsalgorithmen.

Wir können zwei Grade der Datenabstraktion unterscheiden:
- Abstrakte Datenobjekte
- Abstrakte Datentypen

Funktionale Abstraktion

Bei der funktionalen Abstraktion wird lediglich der Algorithmus abstrahiert. In den meisten Programmiersprachen wird solch eine Abstraktion auf eine Funktion abgebildet. Diese bekommt als Parameter die Datenstrukturen, die sie zu bearbeiten hat. Die Lebensdauer dieser endet allerdings mit dem Zurückkehren der Funktion (interne Parameter).

Abstrakte Datenobjekte

Ein abstraktes Datenobjekt erlaubt es, Datenstrukturen und Zugriffsoperationen auf diese Datenstruktur zu einer Einheit zusammenzufassen. Die Lebensdauer der Datenstruktur geht über die das Aufrufende der Zugriffsfunktionen hinaus. Dies ist ein wesentlicher Unterschied zu der funktionalen Abstraktion.

Der Anwender kann nur über die Zugriffsoperationen auf die Datenstruktur zugreifen. Aus der Sicht des Anwenders besteht die Abstraktion darin, daß die Details der Datenstruktur verborgen sind. Ein direkter Zugriff auf die internen Daten ist nicht erlaubt.

Ein Modul, das ein abstraktes Datenobjekt realisiert, nennt man Datenobjekt-Modul. Es besitzt folgende Eigenschaften:

- Passiver Charakter
- Verwaltet die Ablage der internen Daten (Gedächtnis).
- Die Lebensdauer des Gedächtnis beträgt entweder die komplette Laufzeit des Programmsystems oder es wird durch abspeichern dauerhaft aufbewahrt (zB in einer Datei).
- Gedächtnis ist von außen nicht sichtbar (Geheimhaltungsprinzip).
- Daten und Zugriffsoperationen werden in dem Modul zu einer Einheit zusammengefaßt (Verkapselung).
- Zugriffsoperationen sind funktionale Abstraktionen.
- Die Definition der Daten ist gleichzeitig die Deklaration für genau ein Exemplar.

Ein Datenobjekt-Modul entspricht weitgehend einem Objekt (nicht Klasse) in der Objektorientierten Programmierung.

Folgende Abbildung zeigt den prinzipiellen Aufbau eines abstrakten Datenobjekts:

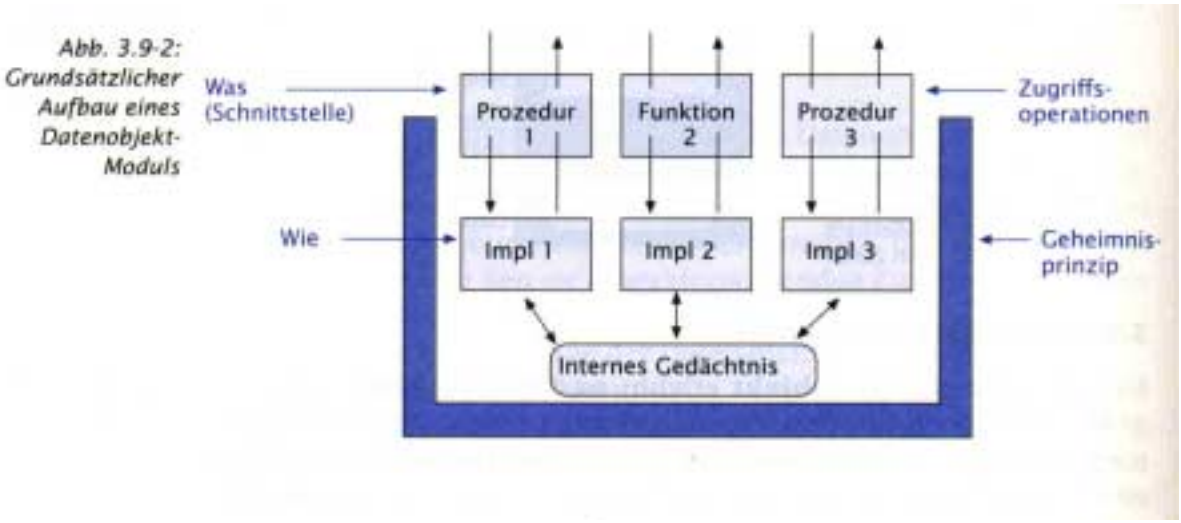
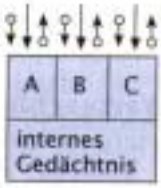
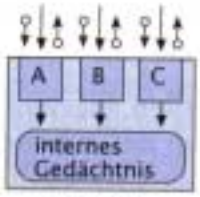
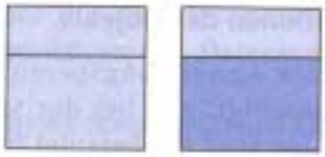
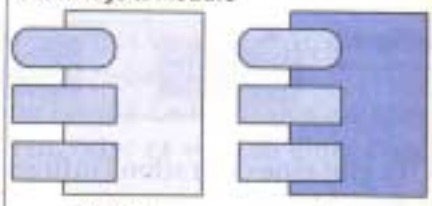


Abb. 3.9-2: Grundsätzlicher Aufbau eines Datenobjekt-Moduls

Es gibt eine Reihe von grafischen Notationen für abstrakte Datenobjekte, die in folgender Tabelle zusammenfaßt werden:

Tab. 3.9-1: Vergleich von Notationen für abstrakte Datenobjekte

/Page-Jones 88/	/Buhr 84/	/Booch 87/
 *informational cluster*	 *package*	funktionale Module  *subprogram specification* *subprogram body*
■ Zusätzlich zu Strukturdiagrammsymbolen.	■ Zusätzlich zu Strukturdiagrammsymbolen. ■ Die Symbole sind schachtelbar. ■ Es gibt extra Symbole für nebenläufige Module. ■ Die Operationen können auch an die Seiten gezeichnet werden.	Datenobjekt-Module  *package specification* *package body*  steht für den Export von Objekten und Typen  steht für den Export von Operationen ■ Aufrufpfeile zeigen nur auf Module, nicht auf Operationen. Datenflüsse werden **nicht** angegeben. ■ Die Notation ist alternativ zu Strukturdiagrammen. ■ Die Symbole sind schachtelbar. ■ Es gibt extra Symbole für nebenläufige Module und abstrakte Datentypen. ■ Die Terminologie ist auf Ada zugeschnitten.

Die Implementierung in den verschiedenen Programmiersprachen verläuft unterschiedlich. In C++ werden sie nicht direkt unterstützt. Es ist aber möglich, das Objekt auf ein .C-Modul abzubilden. Das interne Gedächtnis und die Funktionen, die zur internen Implementierung verwendet werden, werden als static markiert, wodurch sie modulglobal bleiben. Nur die Schnittstellenfunktionen werden exportiert.

Typische Anwendungsgebiete von abstrakten Datenobjekten sind:
- Containerobjekt (Abspeicherung von primitiveren Datenstrukturen)
- Abbildung realer Objekte (zB Roboterhand, Atomreaktor)

Die Verwendung von abstrakten Datenobjekten bringt folgende Vorteile mit sich:
- Benutzung kann alleine durch die Anwendung der Zugriffsoperationen erfolgen. Es ist nicht notwendig, die interne Implementierung zu kennen.
- Durch die Datenabstraktion wird das WAS vom WIE getrennt.
- Zugriffsoperationen können ohne Rücksicht auf das Gedächtnis festgelegt werden, so daß der Anwender das Objekt komfortable benutzen kann.
- Eine Änderung der Implementierung oder des Gedächtnisses zieht keine Änderung des benutzenden Anwendungsprogrammes nach sich.
- Der Anwender muß keine Funktionen zum Manipulieren der Daten schreiben.

Abstrakte Datentypen

Bei abstrakten Datenobjekten wird die gesamte Implementierung auf ein einziges Objekt ausgerichtet. Falls zB eine Roboterhand abstrahiert wird und der Anwender benötigt aber nicht ein Roboterhand-Objekt sondern zwei, müßte er die gesamte Beschreibung (bei C das .C-Modul) textuell kopieren und vervielfältigen.

Das ist kein Problem, wenn ohnedies nur ein Objekt dieser Art im Programm vorkommt. ZB gibt es in der Steuersoftware für ein Atomkraftwerk nur einen Atomreaktor. Wenn allerdings eine gebräuchlicheres Objekt wie zB ein Container-Objekt implementiert wird, das einige hundert Male im Programm vorkommt, ist diese Methode nicht praktikabel.

Deswegen wird das Konzept des abstrakten Datenobjekts verallgemeinert. Dies geschieht dadurch, daß sich die Beschreibung nicht gleich auf ein Objekt bezieht, vielmehr wird ein Objekt-Typ beschrieben, nach dessen Muster dann beliebig viele Instanzen von Objekten gebildet werden können. Man nennt diese Typen abstrakte Datentypen. Sie sind das Äquivalent zur Klasse in der objektorientierten Programmierung.

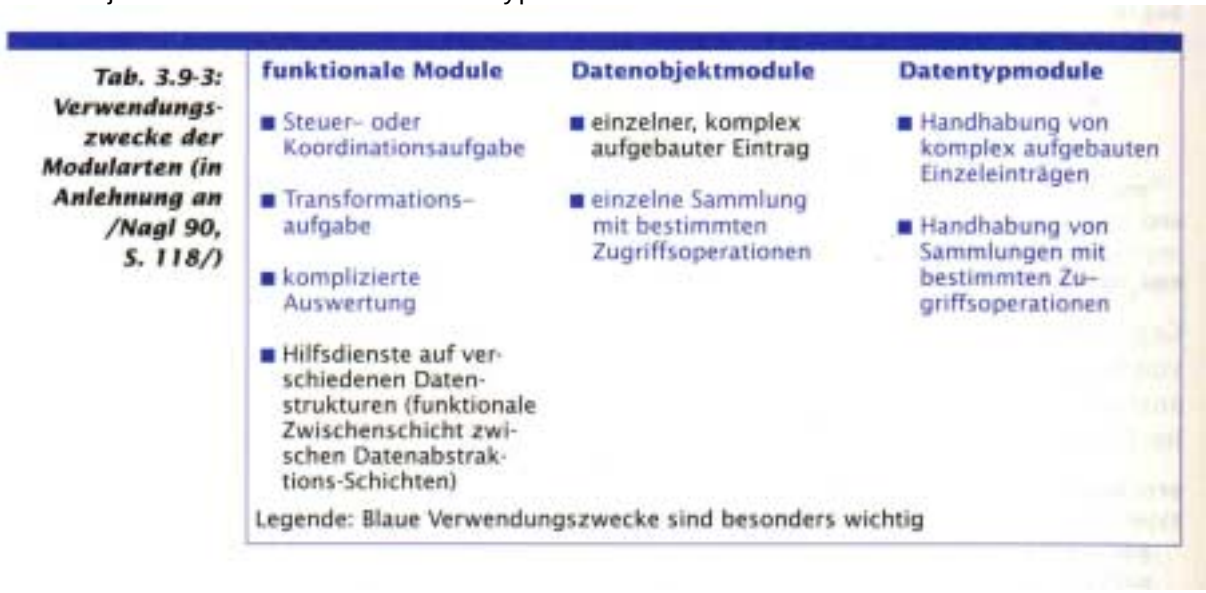
Auf welches Objekt sich die Schnittstellenfunktionen beziehen soll, wird durch diverse Notationen ausgedrückt. In C++ geschieht dies durch die Syntax *Objekt.Zugriffsfunktion*.

Manchmal ist es notwendig, bestimmte Dinge in abstrakten Datentypen zu parametrisieren. ZB möchte der Anwender bei dem abstrakten Datentyp Stack festlegen, welche Datentypen in ihm gespeichert werden sollen. Mann nennt abstrakten Datentypen, die dies erlauben, generische abstrakte Datentypen. In C++ kann diese Parametrisierung über Templates erfolgen.

Der Unterschied eines abstrakten Datentyps zu einer Klasse im objektorientierten Sinn besteht darin, daß bei den ADTs keinerlei Vererbung und daraus folgend kein Polymorphismus vorgesehen ist.

Verwendung

Folgende Grafik faßt die Verwendungen von funktionaler Abstraktion, abstrakten Datenobjekten und abstrakten Datentypen zusammen.



Tab. 3.9-3: Verwendungszwecke der Modularten (in Anlehnung an /Nagl 90, S. 118/)

funktionale Module	Datenobjektmodule	Datentypmodule
■ Steuer- oder Koordinationsaufgabe	■ einzelner, komplex aufgebauter Eintrag	■ Handhabung von komplex aufgebauten Einzeleinträgen
■ Transformations-aufgabe	■ einzelne Sammlung mit bestimmten Zugriffsoperationen	■ Handhabung von Sammlungen mit bestimmten Zugriffsoperationen
■ komplizierte Auswertung		
■ Hilfsdienste auf verschiedenen Datenstrukturen (funktionale Zwischenschicht zwischen Datenabstraktions-Schichten)		

Legende: Blaue Verwendungszwecke sind besonders wichtig