Motivation

» Basierend auf den im ersten Kapitel eingeführten Konzepten für einfache Datenobjekte, die wir als atomare oder elementare Datenobjekte bezeichnen, wird in diesem Kapitel erörtert, dass man mit diesen elementaren Datenobjekten alleine in der Praxis nicht auskommt. Für die algorithmische Lösung komplexerer Aufgabenstellungen sind sowohl Sammlungen von Datenobjekten gleichen Typs als auch solche unterschiedlichen Typs erforderlich, dazu werden die so genannten Felder (arrays) und Verbunde (compounds, structures oder records) eingeführt.

Den bis hierher diskutierten Datentypen (inklusive Feldern und Verbunden) ist gemeinsam, dass Variablen dieser Typen während ihrer Lebensdauer zwar ihren Wert, nicht aber ihre Struktur ändern können. Für die Lösung vieler Aufgabenstellungen benötigt man jedoch Datenobjekte, die nicht nur ihren Wert, sondern auch ihre Größe und u. U. sogar ihre Struktur ändern, die also insbesondere dynamisch wachsen und schrumpfen können – davon rührt auch die häufig verwendete Bezeichnung dynamische Datenstrukturen (dynamic data structures). Solche Datenobjekte sind Sammlungen mehrerer „kleinerer" Datenobjekte, die untereinander in Beziehung stehen, also vernetzt sind, weshalb wir für sie auch den Begriff vernetzte Datenstrukturen (linked data structures) verwenden. In diesem Kapitel führen wir die zur Realisierung solcher flexiblen Datenobjekte erforderlichen Konzepte und Konstrukte ein und erörtern die Besonderheiten von und den Umgang mit wichtigen Repräsentanten solcher Datenstrukturen, insbesondere werden verkettete Listen, Bäume und binäre (Such-)Bäume ausführlich beschrieben.

Das wichtigste Mittel zur Meisterung der Komplexität ist die Abstraktion. Das gilt sowohl für den Entwurf von Algorithmensystemen (in Kapitel 2 behandelt), als auch für die Modellierung komplexer Datenstrukturen. Deshalb führen wir in diesem Kapitel auch Konzepte zur Konstruktion und Verwendung abstrakter Datenstrukturen (abstract data structures), auch Datenkapseln (data capsules) genannt, und abstrakter Datentypen (abstract data types) ein und widmen diesen wegen ihrer Bedeutung jeweils einen eigenen Abschnitt. «

Ziele

Nach dem Lesen dieses Kapitels haben Sie Folgendes erreicht:

- Sie wissen, dass für die algorithmische Lösung komplexer Aufgaben analog zur Gestaltung der Ablaufstrukturen von Algorithmen auch entsprechende Konzepte und Konstrukte zur Modellierung der in den Algorithmen benötigten Datenobjekte erforderlich sind.
- Sie wissen, dass wir zwischen elementaren und strukturierten Datenobjekten unterscheiden, und kennen die Unterschiede zwischen diesen beiden Kategorien von Datenobjekten und den Datentypen, die zu ihrer Realisierung benötigt werden.
- Sie wissen, dass wir zudem zwischen Datenobjekten, bei denen im Zuge der Algorithmenausführung nur ihr Wert verändert werden kann (sie werden hier in elementare und strukturierte Datenobjekte unterteilt) und solchen, bei denen sich neben ihrem Wert auch ihre Größe und/oder ihre Struktur ändern kann (vernetzte Datenobjekte), unterscheiden.
- Sie wissen, was man unter Feld, Verbund, verkettete Liste, Baum, Binärbaum und binärer Suchbaum versteht, kennen Konzepte und Konstrukte zu deren Realisierung, wissen wozu man solche Datenobjekte benötigt und kennen wichtige Standardoperationen zu ihrer Erzeugung und Manipulation.
- Sie wissen, was man unter einer abstrakten Datenstruktur, einer Datenkapsel und einem abstrakten Datentyp versteht, wozu man diese benötigt und wie man sie realisieren kann.

3.1 Atomare Datenobjekte und -typen

In Kapitel 1 haben wir (neben Anderem) grundsätzliche Eigenschaften von Algorithmen erörtert und darauf aufbauend elementare Konzepte und Konstrukte zur Gestaltung derselben erläutert. In diesem Zusammenhang haben wir auch bereits einige elementare Konstrukte zur Definition von Datenobjekten eingeführt. Wir haben dabei zunächst unterschieden zwischen

1. *unveränderbaren* Datenobjekten in Form von Literalen (z.B. 17) oder Konstanten (z.B. *pi*) und
2. *veränderbaren* Datenobjekten, also Variablen (z.B. *x*), die während der Ausführung des Algorithmus, dessen Bestandteil sie sind – vor allem durch die Zuweisungsanweisung –, unterschiedliche Werte aus einer bestimmten Wertemenge (die durch ihren im Zuge der Deklaration festgelegten Datentyp bestimmt ist) annehmen können.

Eine weitere, von dieser Unterscheidung unabhängige, also zu ihr orthogonale Unterscheidungsmöglichkeit ergibt sich aufgrund der den Datenobjekten zugeordneten Datentypen, die neben einer Wertemenge auch die Menge der möglichen Operationen definieren.

In dieser Hinsicht haben wir in Abschnitt 1.3.1 bereits Datenobjekte eingeführt, die ganze Zahlen (Datentyp *integer*), reelle Zahlen (Datentyp *real*), Wahrheitswerte (Datentyp *boolean*) und Zeichen (Datentyp *char*) repräsentieren. Diesen Datenobjekten ist gemeinsam, dass sie in unserem Sinne als logisch ganze, nicht mehr weiter zerlegbare Artefakte aufgefasst werden und in diesem Sinne als atomar angesehen und deshalb auch als *atomare*, *elementare*, *einfache* oder *unstrukturierte* Datenobjekte (bzw. Datentypen) bezeichnet werden, wobei sich keine dieser Bezeichnungen gegenüber den anderen klar durchgesetzt hat.

Wir haben zwar auch schon Texte bzw. Zeichenketten (Datentyp *string*) kurz behandelt, diese zählen wir aber nicht mehr zu den elementaren Datenobjekten. Wir widmen uns den Zeichenketten weiter hinten bei der Behandlung von Feldern noch einmal ausführlicher.

Analog zur Gestaltung der Ablaufstruktur von Algorithmen benötigt man bei der algorithmischen Lösung komplexer Aufgaben auch Konzepte und Konstrukte zur Gestaltung (wir sagen dazu auch *Modellierung*) der zu manipulierenden Datenobjekte. Neben den elementaren Datenobjekten sind dafür vor allem die so genannten *strukturierten* und *dynamischen* oder *vernetzten* Datenobjekte notwendig. In den folgenden Abschnitten behandeln wir Konzepte und Konstrukte für solche Datenobjekte und -typen mit dem Ziel, möglichst adäquate Mittel für eine realitätskonforme Modellierung der in Algorithmen benötigten Daten bereitzustellen.

3.2 Strukturierte Datenobjekte und -typen

In vielen Algorithmen ist es sinnvoll, wenn nicht gar notwendig, eine Menge einzelner Datenobjekte in einer Variablen zusammenzufassen, weil sie z.B. logisch zusammen gehören und daher eine logische (wenn auch strukturierte) Einheit bilden. Bei derartigen Datenobjekten unterscheiden wir zwischen solchen, deren einzelne Elemente alle denselben Datentyp haben – wir nennen diese ***Felder*** (*arrays*) – und solchen, deren Elemente unterschiedliche Datentypen haben können – wir nennen diese ***Verbunde*** (*compounds*).

3.2.1 Felder

Bevor wir unterschiedliche Ausprägungen von Feldern im Detail behandeln, wollen wir eine Definition für dieses Konzept geben.

Definition: *Feld*

Ein *Feld* (*array*) ist ein Datenobjekt, das eine Sammlung einer fixen Anzahl von Elementen repräsentiert, die alle denselben Datentyp haben und auf die über einen *Index* zugegriffen werden kann.