Fachbericht

Thema:

Beschreiben Sie dynamische Datenstrukturen und ihre Einsatzgebiete. Vergleichen Sie mit statischen Datenstrukturen.

Gliederung:

1) Einleitung
2) Statische Datenstrukturen
   - 2.1) Atomare Datentypen
   - 2.2) Fundamentale Datenstrukturen
3) Dynamische / höhere Datenstrukturen
   - 3.1) Liste
   - 3.2) Schlange (Queue)
   - 3.3) Stapel (Stack)
   - 3.4) Bäume
4) Schluß

Autor: Andreas Vennemann

1) Einleitung

Bei der Programmierung werden Probleme der realen Welt in Daten und Programme der Rechnerwelt umgesetzt. Man muß sich also Gedanken über den Algorithmus und die Datenstruktur machen, die man einsetzen möchte, um dieses Problem umzusetzen.
Man unterscheidet grundlegend zwischen dynamischen und statischen Datenstrukturen.

2) statische Datenstrukturen

Statische Datenstrukturen oder Datentypen zeichnen sich dadurch aus, daß sie zwar ihren Wert, nicht aber ihre Struktur zur Laufzeit eines Programms ändern können. Die Größe des benötigten Speichers wird nur einmal bei der Erstellung (Compilierung) des Programmes festgelegt und ist dann während der ganzen Laufzeit gleich.

2.1) atomare Datentypen

Unter atomaren Datentypen versteht man elementare Datentypen mit einem fest definierten Wertebereich, die sich nicht weiter zerlegen lassen (atomar). Die Wertemenge von atomaren Datentypen bezeichnet man auch als Kardinalität dieses Datentyps (Mächtigkeit der zugeordneten Wertemenge).
So haben z.B.: Integer – Variablen eine Teilmenge der ganzen Zahlen als Wertebereich, Charakter die Menge der Schriftzeichen, Boolean Wahrheitswerte (true und false) und Zeiger den Adreßbereich als Kardinalität.

2.2) fundamentale Datenstrukturen

Fundamentale Datenstrukturen bestehen aus einer oder mehreren Komponenten, welche atomare Datentypen sind. Weiterhin wird für die Datenstruktur eine Beziehung zwischen den einzelnen Komponenten definiert. (z.B.: 1:1 => Liste; 1:n => Baum; n:n => Netz oder Graph)
Fundamentale Datenstrukturen sind z.B.: Felder (Arrays). Sie bestehen aus einer festen Anzahl zusammenhängender Datentypen gleicher Art. Man bezeichnet Felder daher auch als homogene Datenstruktur. Die 1:1 Beziehung zwischen den Komponenten wird durch einen Index abgebildet. Über eine Funktion wird aus dem Index eines Feldes intern die Speicheradresse berechnet.
Ein weiterer fundamentaler Datentyp sind Sätze (Strukturen oder Records). Sie bestehen aus mehreren unterschiedlichen atomaren Datentypen, wobei die einzelnen Komponenten über den Namen der Struktur und den Namen der Komponente angesprochen werden. (Beispiel in C: Adresse.Vorname oder Adresse=>Vorname) Man bezeichnet Sätze auch als heterogene Datenstruktur.

Autor: Andreas Vennemann

3) Dynamische / höhere Datenstrukturen

Dynamische Datenstrukturen können während der Laufzeit ihren Wert und ihre Struktur (und somit ihre Größe) verändern. Durch diese Tatsache kann einer dynamischen Datenstruktur jedoch kein fester Speicherplatz zugeordnet werden. Der Speicher muß also während der Laufzeit zugewiesen werden, sobald er benötigt wird. Als strukturelles Hilfsmittel dienen Zeiger, welche die einzelnen Komponenten mit einander verbinden (verpointern). Dynamische Datenstrukturen besitzen also immer einen Datenteil und einen Zeiger auf eine andere Komponente.

3.1) Liste

Die Komponenten einer Liste heißen Knoten und enthalten einen Datenteil und einen oder mehrere Zeiger auf weitere Knoten. Man kann also z.B.: einen Zeiger auf den Nachfolgeknoten, einen Zeiger auf den Vorgängerknoten usw. definieren. Dabei muß man abwägen, ob sich dieser zusätzliche Aufwand in der Implementierung im Vergleich zum Einsatz der Liste auch lohnt. (z.B.: kann die Komplexität eines Algorithmus sinken, wenn man einen weiteren Zeiger hinzufügt.)
Strukturell entsteht eine sequentielle Folge von Knoten.
Listen werden z.B. eingesetzt, um sehr große Zahl darzustellen. Die Zahl wird dazu als Polynom aufgefaßt, wobei in den Knoten der Liste der Exponent zur Basis 10 und der Wert an dieser Stelle gespeichert wird.

3.2) Schlange (Queue)

Die Datenstruktur Schlange ist eine Liste, bei der nur am Listenende Daten eingefügt und am Listenanfang Daten entnommen werden dürfen. D.h. also das älteste Element wird als erstes bearbeitet (gelöscht etc.). Diese Eigenschaft bezeichnet man als FIFO – Verhalten (First – In – First – Out). Die Reihenfolge der Komponenten bleibt zu jeder Zeit erhalten. Queues können daher immer dann verwendet werden, wenn eine bestimmte Reihenfolge von Daten abgebildet werden soll. (u.a. bei der Verwaltung von Druckaufträgen; die Druckaufträge werden zum Drucker geschickt und der Reihe nach ausgegeben.)

3.3) Stapel (Stack)

Der Stapel ist ebenfalls eine Liste, wobei Manipulationen (Einfügen, löschen, etc.) nur mit dem letzten Knoten erlaubt sind. D.h. also, das zuletzt eingefügte Element wird auch als erstes wieder abgearbeitet. (LIFO – Verhalten, Last – In – First – Out).
Der Stack wird z.B. bei Funktionsaufrufen verwendet. Erfolgt in einem Programm ein Funktionsaufruf, dann wird die Sprungadresse auf einen internen Stack geschrieben und bei Beendung der Funktion wieder zu dieser Adresse zurück gesprungen.

3.4) Bäume

In einem Baum existiert genau ein Knoten, von dem aus alle anderen Knoten genau einmal erreicht werden können (1:n Beziehung). Diesen “Ausgangsknoten” bezeichnet man als Wurzel des Baumes. Bäume können immer dann verwendet werden, wenn man eine bestimmte Hierarchie abbilden möchte.
Besonders oft werden sog. Binärbäume eingesetzt, wobei in einem Binärbaum jeder Knoten, keinen, einen, oder maximal zwei Nachfolger (Söhne) hat.
In einem binären Suchbaum, bei dem jeder Knotenwert größer ist als alle Knotenwerte seiner Söhne kann eine schnellere Suche (O(Log n)) erfolgen, als in einer Liste (O(n)).

4) Schluß

Man hat also eine Vielzahl von Datenstrukturen zur Auswahl um ein Problem umzusetzen. Man sollte jedoch immer beachten, daß die Laufzeit eines Programms auch immer wesentlich von der eingesetzten Datenstruktur abhängt.

Autor: Andreas Vennemann