2.5. Pointer, Listen

Listen sind Folgen von Elementen des gleichen Datentyps. Sie werden durch *Zeiger (pointer)* realisiert. Die Verkettung kann einfach oder doppelt sein. Das erste und/oder das letzte Element sind von außen über einen Zeiger erreichbar (auch *Anker* der Liste genannt). Der Zugriff erfolgt *sequentiell*; man durchläuft also die Liste von vorne nach hinten bzw. von hinten nach vorne, um nach einem Element zu suchen oder ein Element einzufügen.

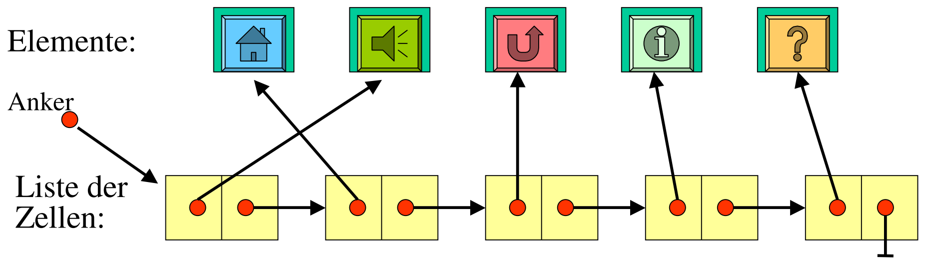
Vgl. Informatik I (Prof. Lagally) und Skript Plödereder Abschnitt 1.5.2

Das Schlüsselwort für Zeiger lautet in Ada ***access***.
Typische Definition einer Liste in Ada:

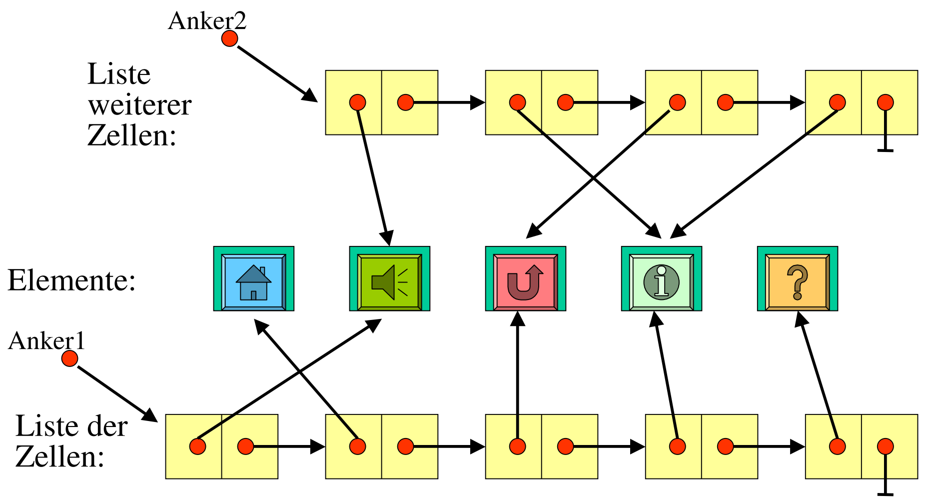
```
type Element is ....;   -- Definiere den Datentyp der Listenelemente
type List_Element;      -- Vorwärtsverweis
type List_Element_Zeiger is access List_Element;
type List_Element is record
                            Inhalt: Element;
                            Next: List_Element_Zeiger;
                        end record;
```

Dies ist eine Liste, in der die Elemente direkt stehen. In der Praxis kann dies lästig sein (vgl. Folien 40-42 über Felder), da ein Element in vielen Listen auftreten kann und dann auch in allen Listen gespeichert und geändert werden muss. Also:

```
type Element is ....;        -- Definiere den Datentyp der Listenelemente
type Element_Zeiger is access Element;
type Zelle;                  -- Vorwärtsverweis
type Zelle_Zeiger is access Zelle;
type Zelle is  record
                  Inhalt: Element_Zeiger;
                  Next: Zelle_Zeiger;
               end record;
```



Will man nun Elemente in mehreren Listen gleichzeitig verwenden, so kann dies ohne Kopien geschehen:



Vorteile dieser Zellen-Darstellung:

- Elemente können in verschiedenen Listen sein.
- Das Ändern von Elementen geschieht synchron überall.
- Das Einfügen in andere Listen ist einfach.
- Es lassen sich weitere Zugriffsstrukturen leicht aufbauen.

Nachteile dieser Zellen-Darstellung:

- Der Zugriff auf Elemente dauert etwas länger.
- Man braucht etwas mehr Speicherplatz.
- Die Speicherverwaltung kann deutlich schwerer werden!

Hinweis: Listen werden in der Halde abgelegt. Nur die Anker stehen im statischen Bereich des Programms, sofern sie deklarieret Variablen sind.

Bitte wiederholen: Informatik I (Prof. Lagally) S. 135 und 138, WS 01/02

2.6. Stapel, Warteschlangen

Stapel und Warteschlangen sind Listen mit speziellen zulässigen Operationen (also (abstrakte) Datentypen).

Stapel oder Stack oder Keller oder Pushdown:

newstack	-- Leeren des Stacks
isempty	-- Abfrage, ob der Stack leer ist
top	-- Oberstes Element im Stack
push	-- Füge ein Element oben an
pop	-- Lösche das oberste Element
length	-- aktuelle Länge des Stacks

Warteschlange oder Queue:

newqueue	-- Leeren des Schlange
isempty	-- Abfrage, ob Schlange leer ist
first	-- Vorderstes Element der Schlange
rest	-- Schlange ohne erstes Element
enqueue	-- Füge Element hinten an
dequeue	-- Entferne erstes Element
length	-- Länge der Schlange

Zu Ringlisten und Doppelschlangen siehe WS 01/02, Einführung in die Informatik I, S. 145 ff

Beispiel 2.3: Implementierung von mehreren Stapeln / Stacks

Aufgabe: Wir wollen eine Multistapelverwaltung in einem eindimensionalen Feld durchführen, d.h.:

Jemand möchte n Stacks verwalten. Insgesamt hat er einen linearen Speicher Sp(1..M) zur Verfügung.

Spontane Idee: Linearer Speicher von 1 bis M, jeder Stack erhält gleichviele Speicherplätze M/n: