

Programmiertechniken 1

Dynamische Datenstrukturen

Learning Outcomes

Sie können ...

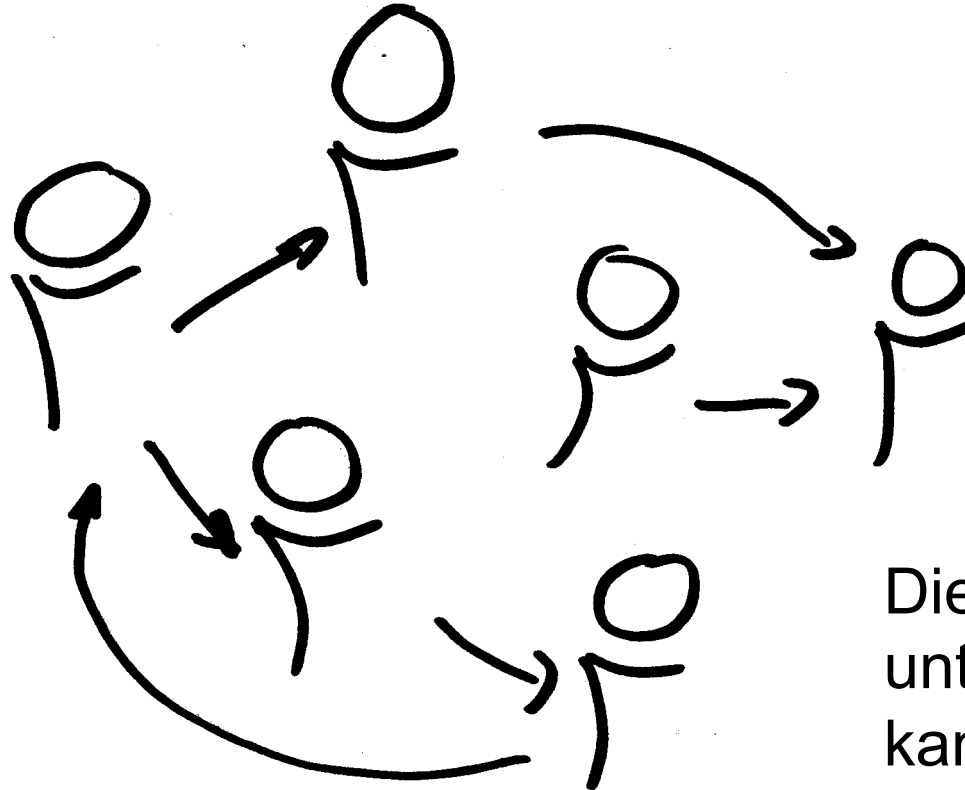
- ... dynamisch Datensätze zu Datenstrukturen aufbauen und modifizieren, d.h.
 - Datenstrukturen entwickeln,
 - Listen der Datenstrukturen erstellen und modifizieren sowie
 - Datensätze in Listen sortieren, indem sie die Datensätze sortiert in eine zweite Liste einfügen.

Dynamische Datenstrukturen

Die Personen
haben Satz an
Daten
(Datensatz) mit
fester Struktur.

→ `struct`

→ `typedef`



Die Verbindung
untereinander
kann sich
ändern.

Vorübung

Sie sind in einem stockdunklen Raum gefangen.
Jemand drückt ihnen eine Büroklammer in die eine Hand. „Da hängen noch mehr dran. Die Kette darfst Du nie verlieren!“

„An einer Büroklammer hängt ein Zettel. Finde die Klammer!“

Regeln:

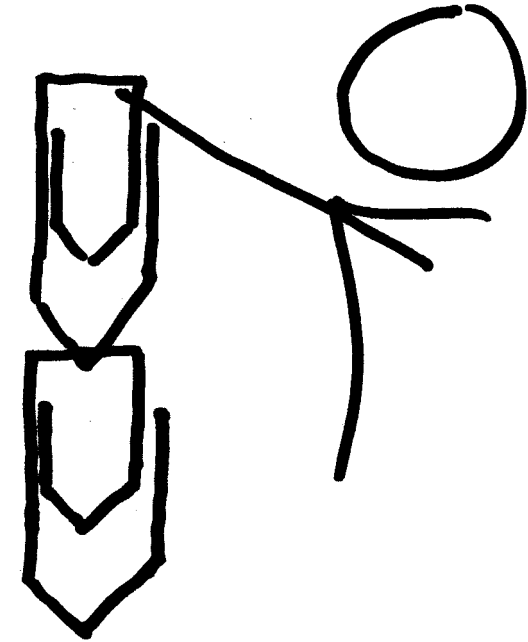
Man darf nur vom Anfang an der Kette folgen.

Man darf fühlen, ob noch eine weitere Büroklammer folgt.

Man muss die Klammern beim Verfolgen berühren.

Frage:

Was machen die Hände im Detail?



Vorübung

Sie sind in einem stockdunklen Raum gefangen.
Jemand drückt ihnen eine Büroklammer in die eine Hand. „Die Kette darfst Du nie verlieren!“ Sie bekommen eine weitere Büroklammer in die andere Hand. „Häng diese an das Ende an!“

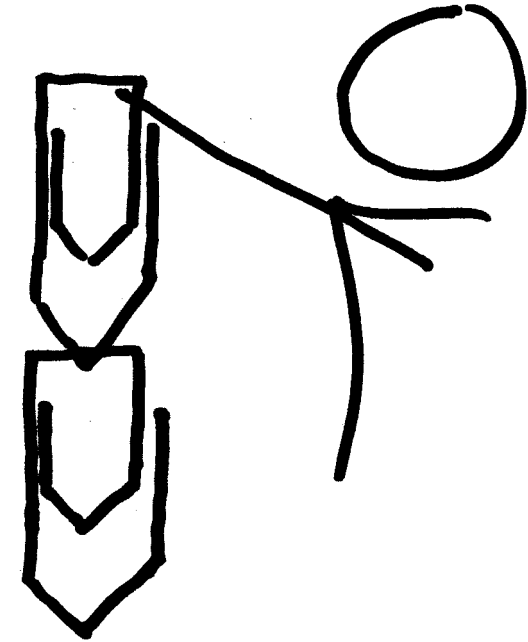
Regeln:

Man darf nur vom Anfang an der Kette folgen.

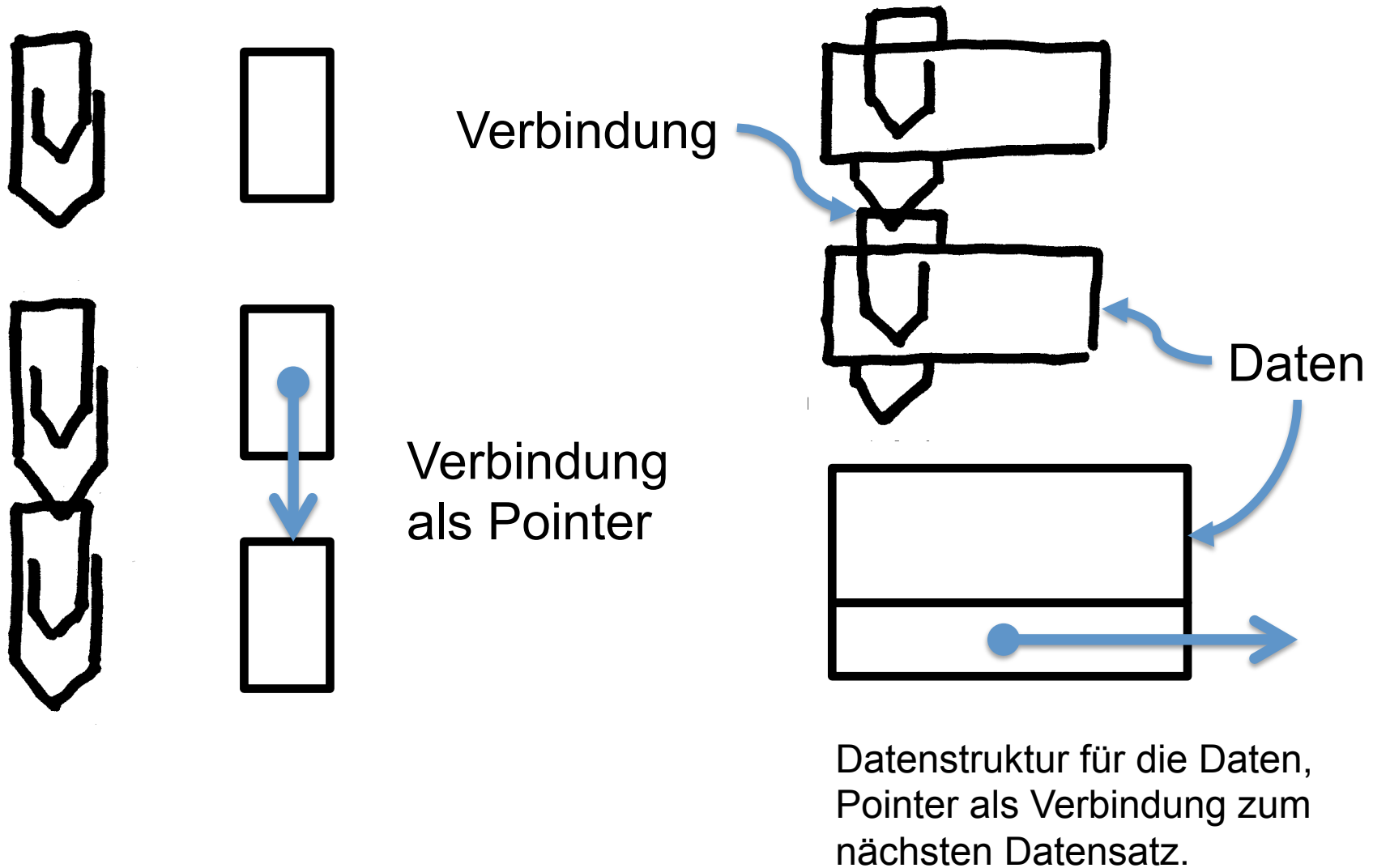
Man darf fühlen, ob noch eine weitere Büroklammer folgt.

Frage:

Welche Büroklammern müssen Sie festhalten, um die Verbindungen herzustellen? Wie viele Hände benötigen Sie dazu eigentlich? Was machen die Hände im Detail?

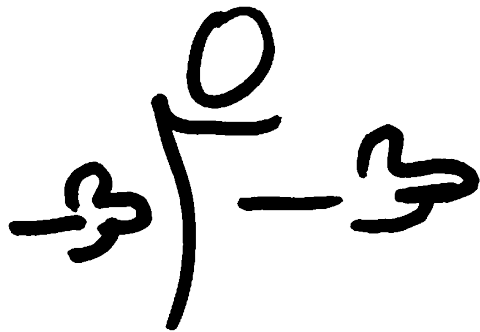


Modellierung mit Datenstrukturen

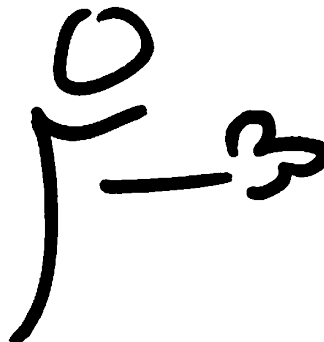


Liste über Pointer

M232212



M232213



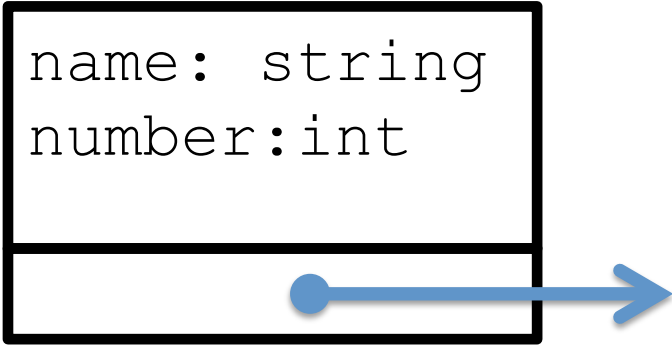
M232214



Datenstrukturen können über Pointer miteinander verkettet werden. Einzelne Datensätze können ebenso über Pointer referenziert werden.

Demo

Code für Studenten erarbeiten



A diagram of a student structure represented as a box. The box is divided into two horizontal sections. The top section contains the text 'name: string' and 'number: int'. The bottom section is empty. A blue arrow points from the right side of the bottom section to the right.

```
name: string  
number: int
```

```
typedef struct student {
```

```
    char name[30];  
    int number;
```

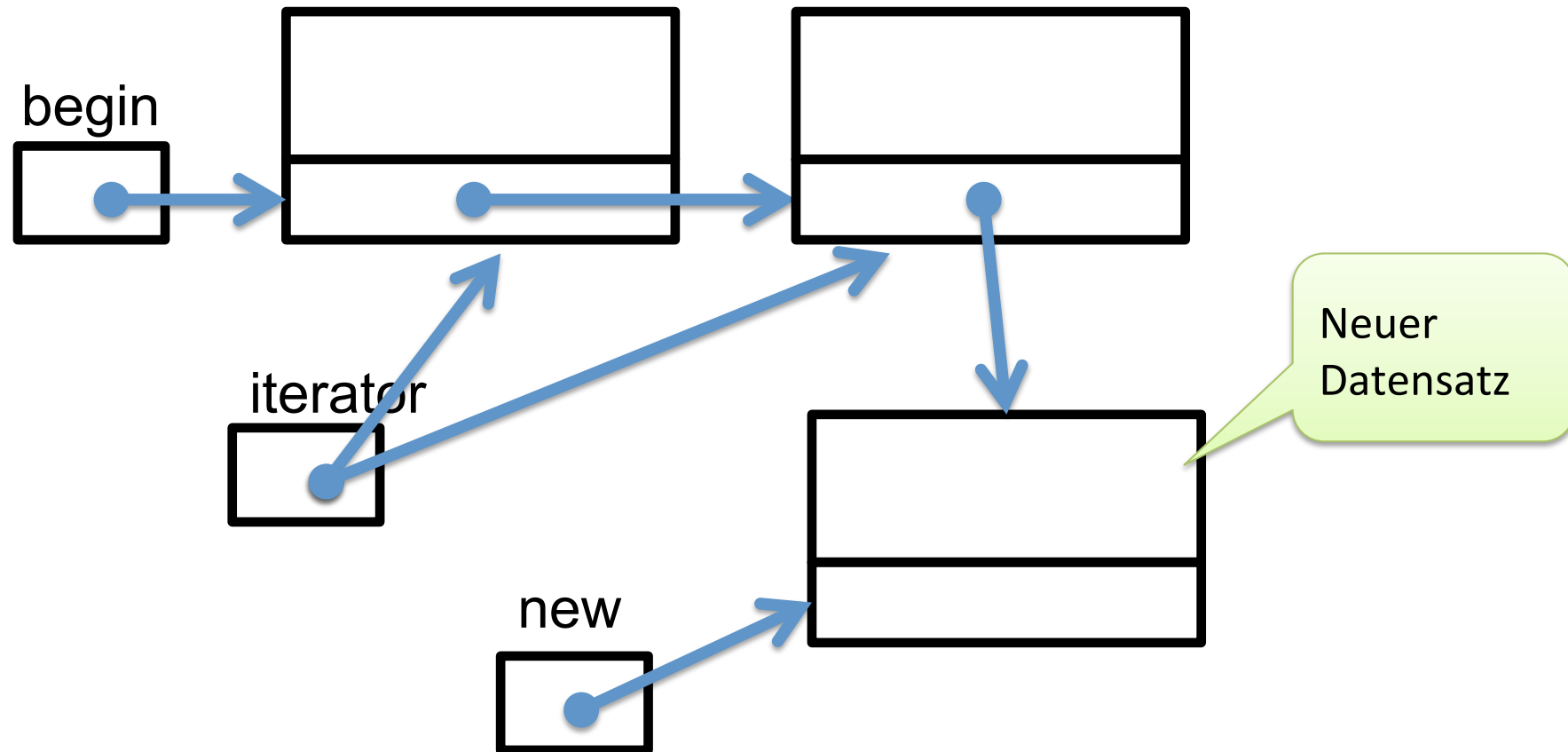
```
    struct student *next;
```

```
} student_t;
```

In der Demo vereinfacht nur die Matrikelnummer.

Anhängen an Liste

Über einen zusätzlichen Pointer (Iterator) wird das Ende gesucht. Dann wird vom letzten Element der Verweis auf das nächste Element auf das neue Element gesetzt.



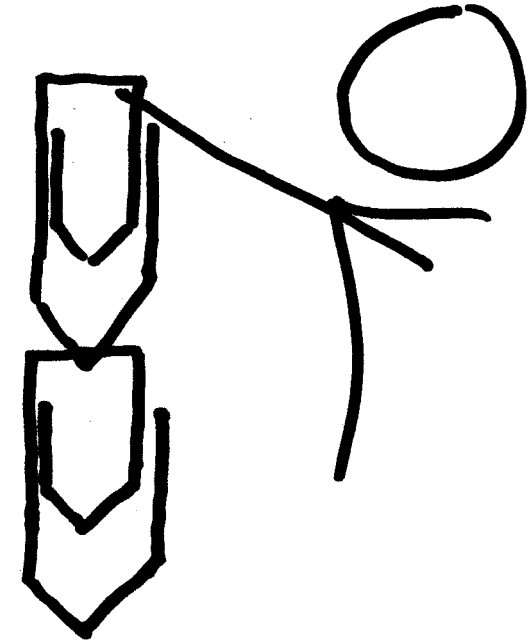
Vorübung

Sie sind in einem stockdunklen Raum gefangen.
Jemand drückt ihnen eine Büroklammer in die eine Hand. „Die Kette darfst Du nie verlieren!“ Sie bekommen eine weitere Büroklammer in die andere Hand. „Häng diese zwischen der 2. und 3. Klammer ein!“

Regeln:

Man darf nur vom Anfang an der Kette folgen.

Man darf fühlen, ob noch eine weitere Büroklammer folgt.

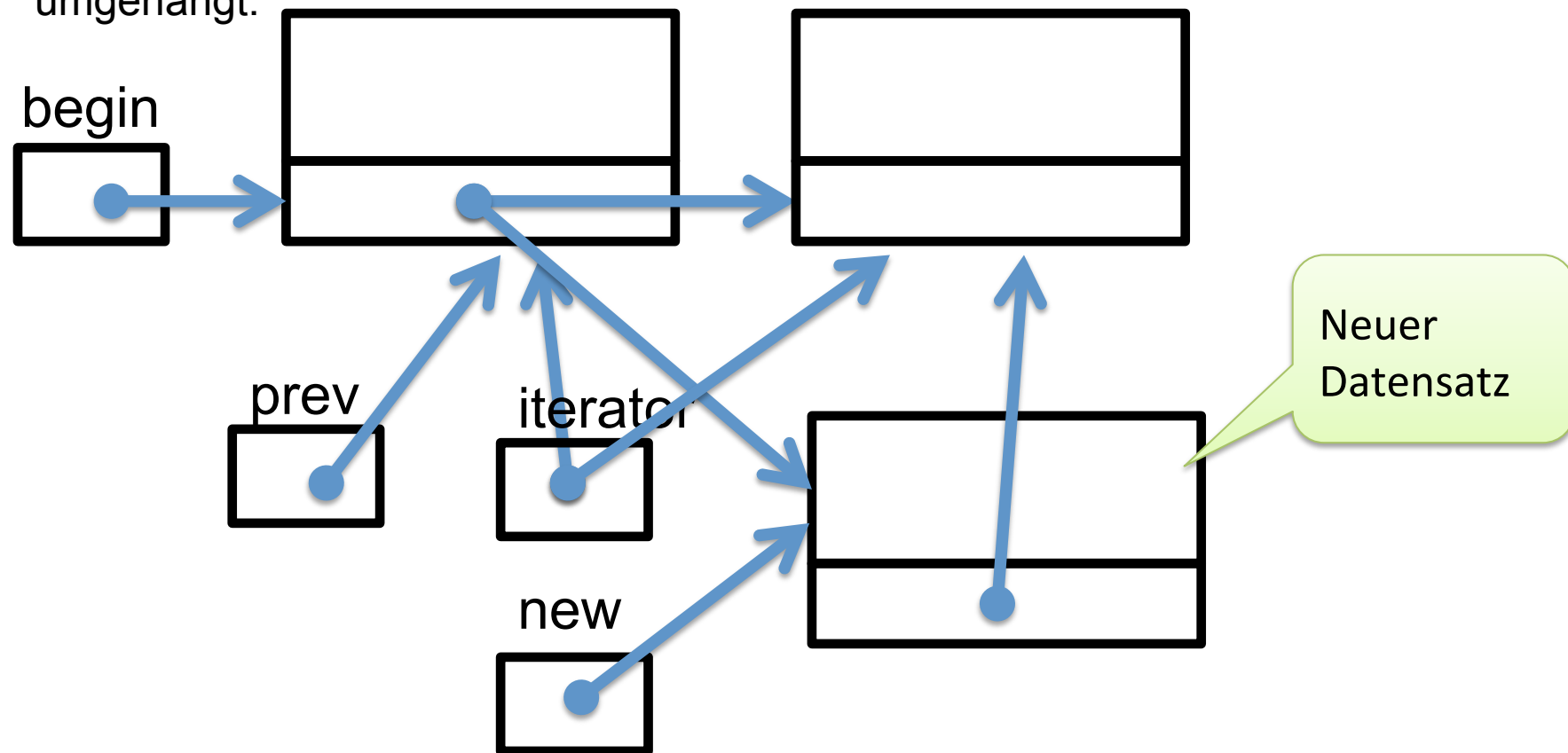


Frage:

Welche Büroklammern müssen Sie festhalten, um die Verbindungen herzustellen? Wie viele Hände benötigen Sie dazu eigentlich? Was machen die Hände im Detail?

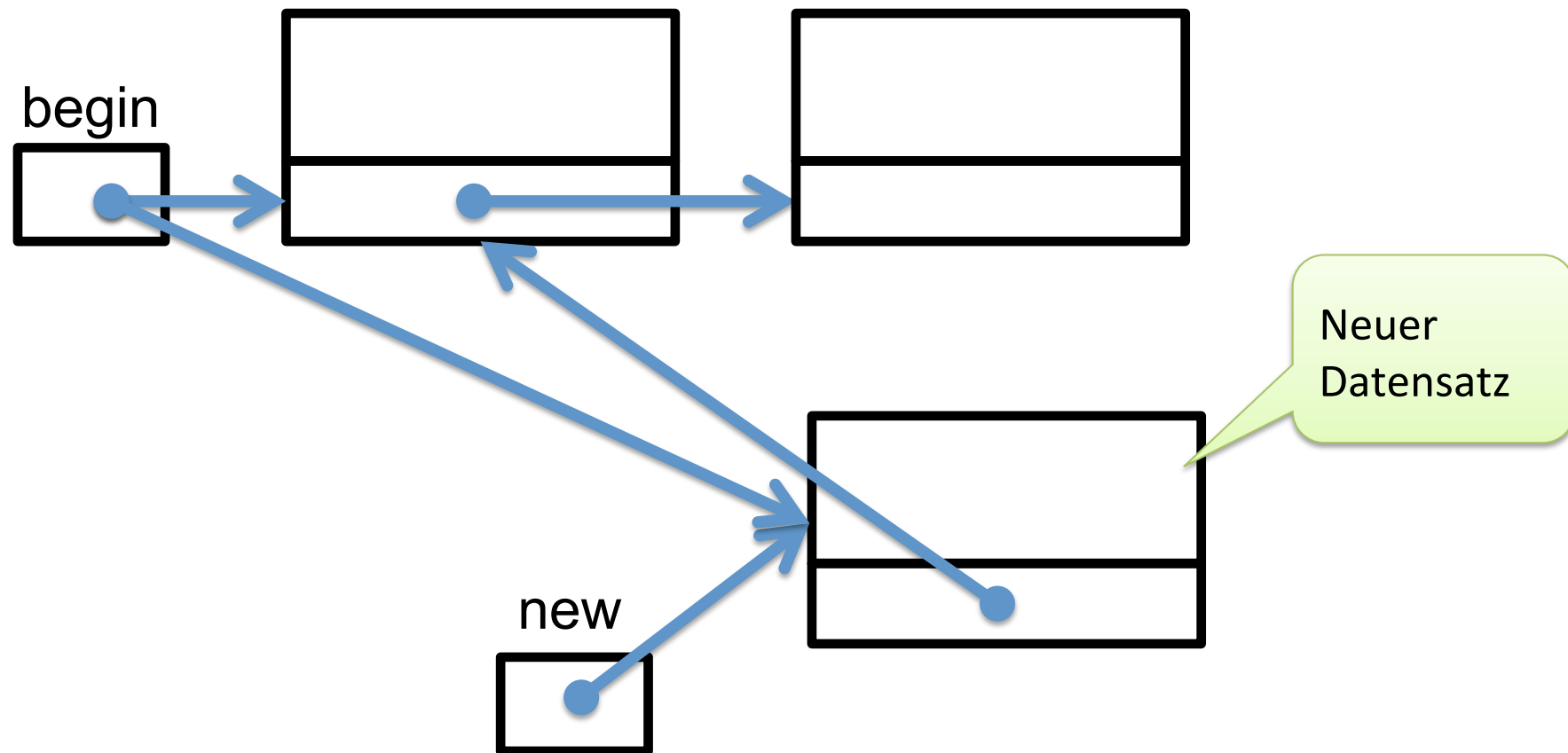
Einfügen in einer Liste

Über einen zusätzlichen Pointer (Iterator) wird die Einfügeposition gesucht (hier nach dem ersten Element). Über einen zweiten Pointer wird die Trennstelle markiert. Dieser zweite Pointer wird mitbewegt. Dann werden die Pointer umgehängt.



Einfügen in einer Liste - Sonderfall

Muss das neue Element vor dem ersten Element eingefügt werden, so muss der Pointer auf das erste Element verändert werden.



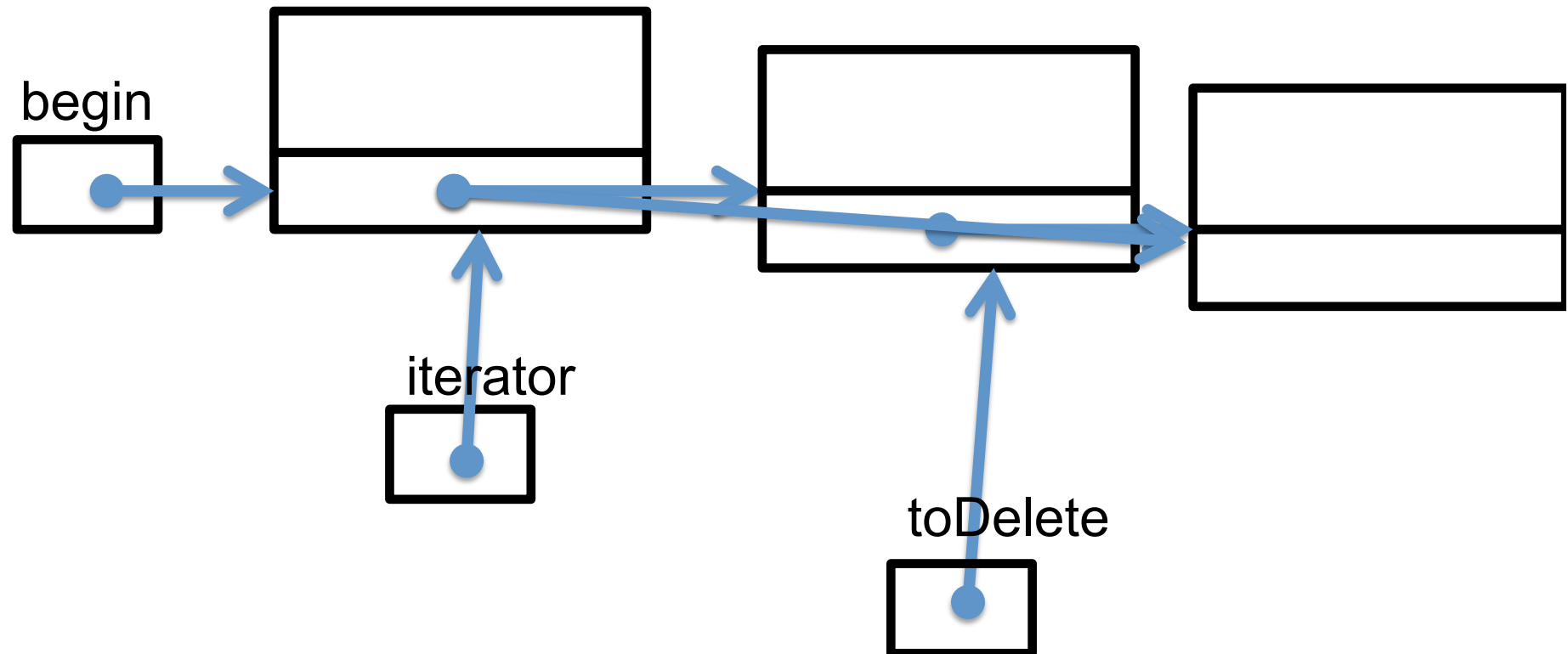
Allgemein Einfügen in eine Liste

Beim Arbeiten mit Listen müssen folgende Fälle beachtet werden:

- Die Liste ist leer.
- Das Einfügen erfolgt vor dem ersten Element
- Das Einfügen erfolgt zwischen zwei Elementen
- Das Einfügen erfolgt nach dem letzten Element (anhängen).

Löschen aus einer Liste

Ist das zu löschende Element gefunden, so werden die Pointer vom Vorgänger auf das nachfolgende Element umgehängt.



Achtung: Eventuell ist das zu löschende Element das erste Element!

Code example

```
// delete element
iterator = begin;
while(iterator->next != toDelete) {
    iterator = iterator->next;
}
if(iterator == begin) {
    begin = iterator->next;
} else {
    iterator->next = iterator->next->next;
}
free(toDelete);
```

Sortieren von Liste

Es gibt verschiedenste Algorithmen für das Sortieren von Listen (Bubble Sort, Quick-Sort, ..). Für eine aufsteigende Liste ist einer der einfachsten der folgende Algorithmus:

- Finde des kleinste Element in der Liste.
- Hänge das gefundene Element der sortierten Liste an.
- Lösche das gefundene Element aus der ersten Liste.
- Mache weiter bei 1., bis die Liste leer ist.

Achtung: Alle Sonderfälle müssen beachtet werden!