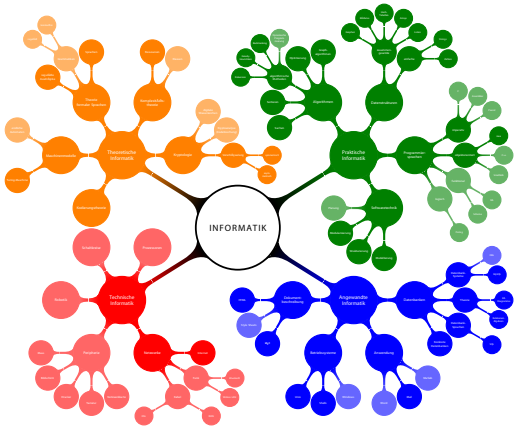
UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR
THEORETISCHE INFORMATIK

Kapitel 26

Listen – Iteration und Modifikation

Von verbogenen Zeigern

Vorlesung Einführung in die Informatik 1 vom 4. Februar 2014 von Till Tantau



Lernziele von Kapitel 26

1. Code zur Iteration über Listen erstellen können
2. Code zur Modifikation von Listen erstellen können

Gliederung von Kapitel 26

Wie besucht man alle Elemente einer Liste?

Problemstellung

Wir wollen alle Elemente (= Zellen) einer Liste »besuchen« und dort »irgend etwas tun«.

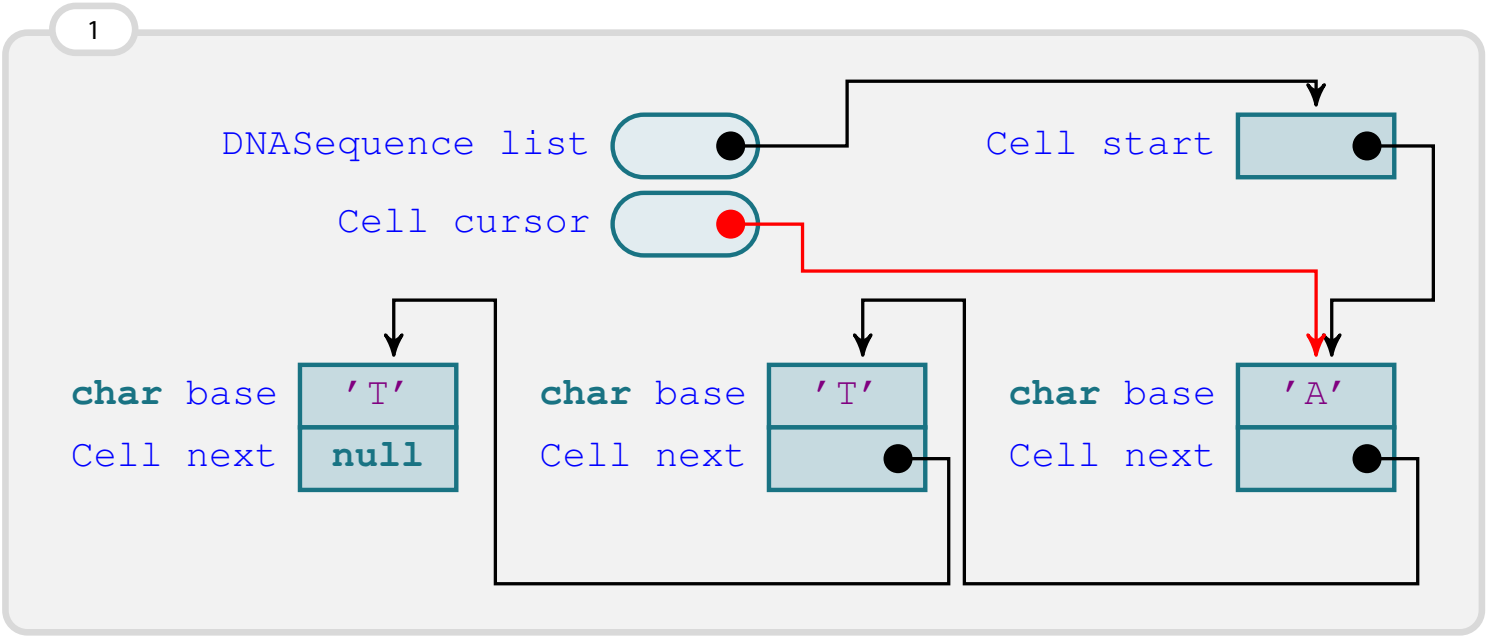
Iterativer Algorithmus

1. Setze `cursor` auf die Startzelle.
2. Tue das Gewünschte für diese Startzelle.
3. Solange `cursor` einen Nachfolger hat, tue:
   - 3.1 Setze `cursor` auf den Nachfolger von `cursor`.
   - 3.2 Tue das Gewünschte für die aktuelle Stelle.

Wer zeigt wann wohin?


```
Cell cursor = list.start; // Startsituation
tue_was (cursor);
while (cursor.next != null) {
  cursor = cursor.next; // Vorwärts!
  tue_was (cursor);
}
```

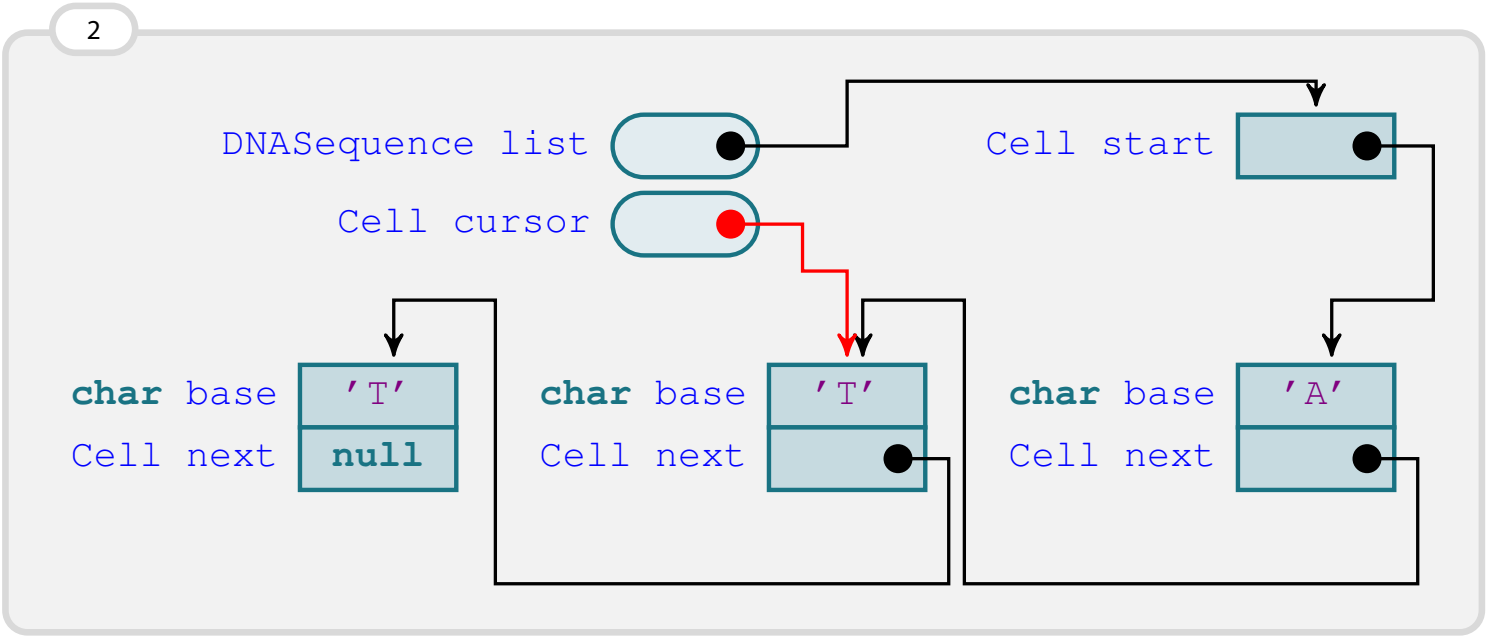
1



Wer zeigt wann wohin?

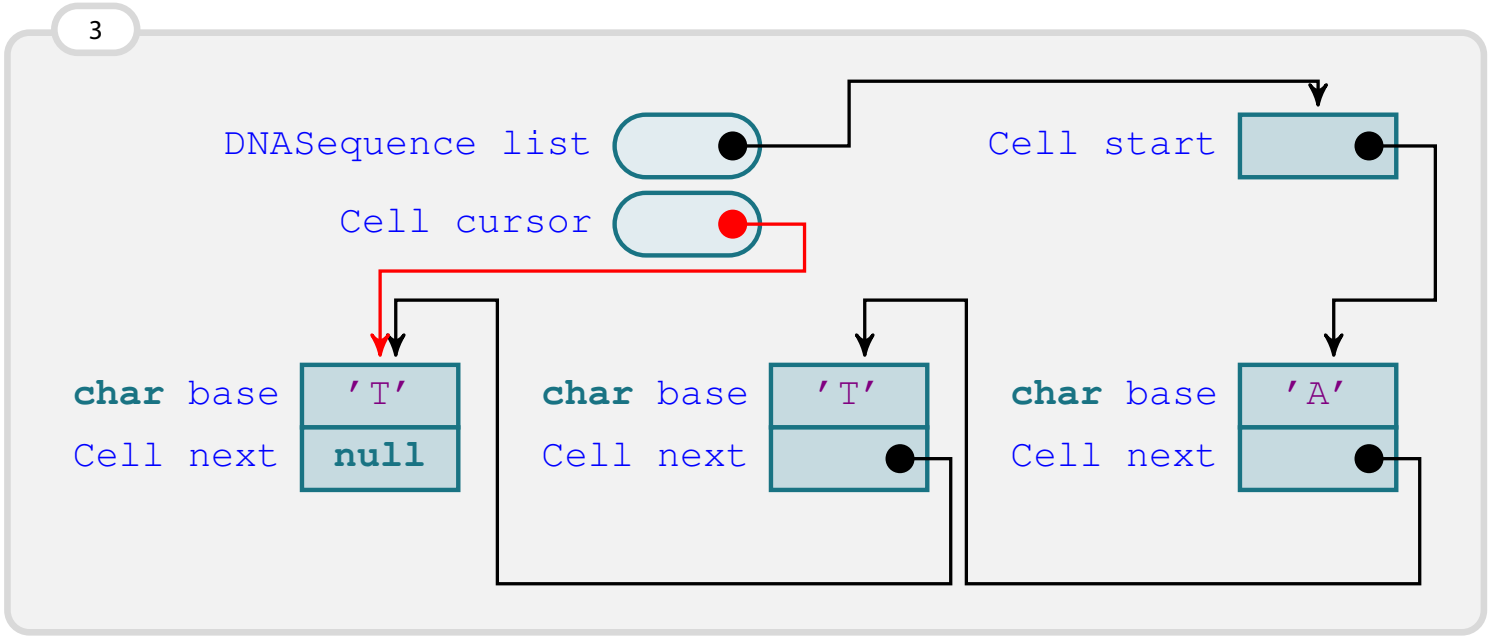

```
Cell cursor = list.start; // Startsituation
tue_was (cursor);
while (cursor.next != null) {
  cursor = cursor.next; // Vorwärts!
  tue_was (cursor);
}
```

2



Wer zeigt wann wohin?


```
Cell cursor = list.start; // Startsituation
tue_was (cursor);
while (cursor.next != null) {
  cursor = cursor.next; // Vorwärts!
  tue_was (cursor);
}
```



Eine effizientere Version des Codes.

Der Code von eben hat noch zwei Nachteile:

1. Der Aufruf von `tue_etwas` steht zweimal im Code. Dies ist unschön, wenn man stattdessen etwas komplexeres machen möchte.
2. Der Code funktioniert nicht, wenn die Liste leer ist, also sofort `cursor == null` gilt.

Diese Probleme lassen sich wie folgt umgehen:

```
Cell cursor = list.start;
while (cursor != null) {
  tue_etwas (cursor);
  cursor = cursor.next;
}
```

Problemstellung: Die Länge einer Liste bestimmen.

- Die Länge einer Liste »sieht man ihr nicht an«.
- Man *muss* alle Elemente einmal besuchen und dabei einen Zähler für jedes besuchte Element hochzählen.
- Die Aktion »tue etwas« ist hier gerade das Hochzählen dieses Zählers.

```
// Algorithmus zum Zählen der Elemente einer Liste
int counter = 0;
Cell cursor = list.start;
while (cursor != null) {
  counter++;
  cursor = cursor.next;
}
```

Die Längenmethode.

- Die Längenberechnung sollte durch eine Methode der Listenklasse implementiert werden.
- Dann kann man ein Listenobjekt erstellen und es später mittels einer Nachricht fragen, was seine Länge ist.

```
class DNASequence {
  // ...

  public int length () {
    int counter = 0;
    Cell cursor = this.start;
    while (cursor != null) {
      counter++;
      cursor = cursor.next;
    }
    return counter;
  }
}
```

5-Minuten-Aufgabe

Zur Übung

Geben Sie den Code einer Methode `count_As`, die die Anzahl an »A«'s in der Liste zurückgibt.

Problemstellung: Ausgabe aller Elemente.

- Um alle Elemente einer Liste auszugeben, muss man sie einfach alle besuchen.
- Die Aktion »tue etwas« ist dann gerade die Ausgabe.

```
class DNASequence {
  // ...
  public void print () {
    Cell cursor = this.start;
    while (cursor != null) {
      System.out.print(cursor.base);
      cursor = cursor.next;
    }
  }
}
```

5-Minuten-Aufgabe

Zur Übung

Geben Sie den Code einer Methode an, die die Basen in der Liste als einen String zurückgibt. Die Idee ist, jede Base nacheinander an das Ende eines Strings anzuhängen:

```
return_me = return_me + cursor.base;
```

Problemstellung: Verändern aller Elemente.

- Wir wollen nun alle Elemente einer Liste verändern, beispielsweise durch ihre Komplemente ersetzen.
- Die Aktion »tue etwas« ist dann gerade diese Modifikation.
- Ein solches Verändern aller Element wird in der funktionalen Programmierung *Map* genannt.

```
class DNASequence {
  // ...
  public void complement () {
    Cell cursor = this.start;
    while (cursor != null) {
      if      (cursor.base == 'A') { cursor.base = 'T'; }
      else if (cursor.base == 'T') { cursor.base = 'A'; }
      else if (cursor.base == 'C') { cursor.base = 'G'; }
      else if (cursor.base == 'G') { cursor.base = 'C'; }
      cursor = cursor.next;
    }
  }
}
```

Problemstellung: Suche nach einem Element.

- Wir wollen nun ein Element finden mit einer bestimmten Eigenschaft; beispielsweise das erste »A«.
- Die Aktion »tue etwas« ist dann der Test, ob der `cursor` eine Zelle mit der gesuchten Eigenschaft erreicht hat.

```
class DNASequence {
  // ...
  public Cell searchForFirstA () {
    Cell cursor = this.start;
    while (cursor != null) {
      if (cursor.base == 'A') {
        return cursor; // Gefunden! Danke und tschüss.
      }
      cursor = cursor.next;
    }
    return null; // Nicht gefunden. Grrr.
  }
}
```

Wie fügt man ein Element in die Mitte einer Liste ein?

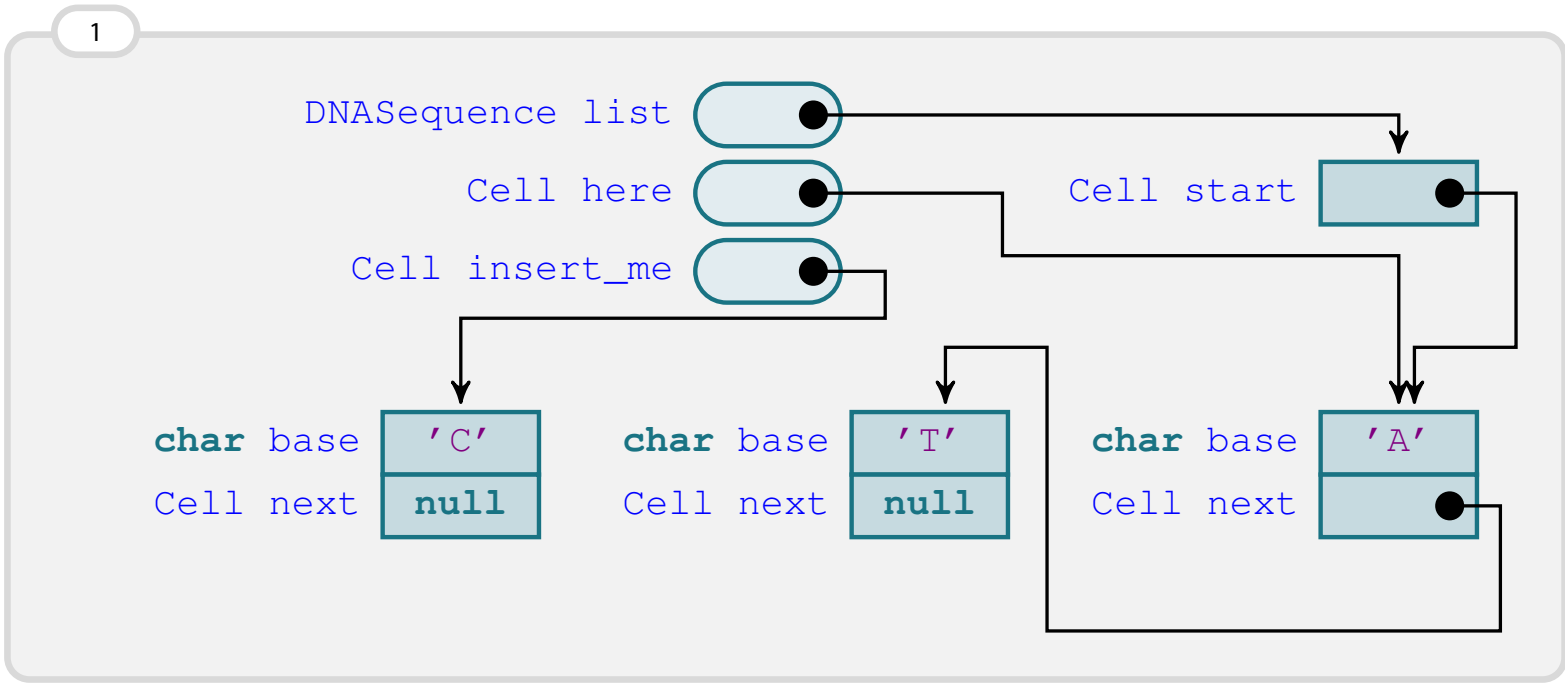
- Wir wollen ein neues Element nicht am Anfang einer Liste, sondern irgendwo zwischendrin einfügen.
- Dazu muss man *eigentlich nur lokal die Verkettung ändern*.
- Genauer braucht man zunächst einen Verweis auf ein Element `a`, *nach dem man* das neue Element `b` einfügen möchte.
- Dann ändert man zwei Verweise:
  - Der Nachfolger von `a` ist nun `b`.
  - Der Nachfolger von `b` ist nun der alte Nachfolger von `a`.

Der Code einer Einfügemethode.


```
class DNASequence {
  // ...
  void insertAfter (Cell here, char b) {
    Cell insert_me = new Cell ();
    insert_me.base = b;
    insert_me.next = here.next;
    here.next      = insert_me;
  }
}
```

Wer zeigt wann wohin?

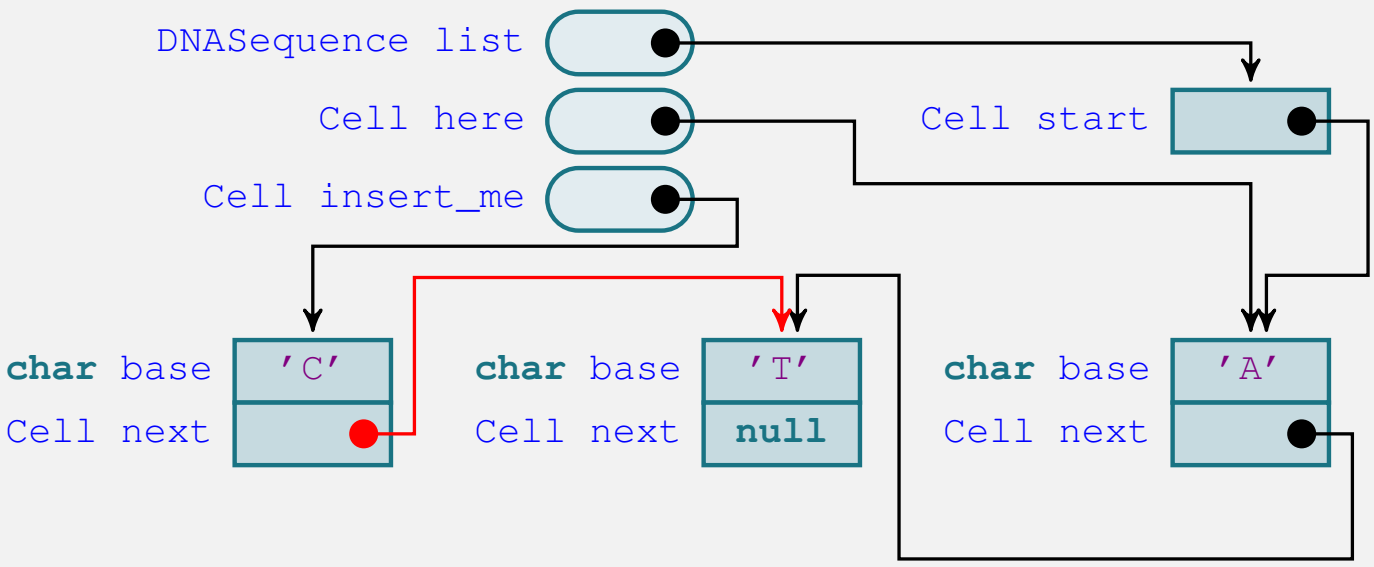

```
                                  // 1. Ausgangssituation
insert_me.next = here.next;       // 2. Erste Veränderung
here.next      = insert_me;       // 3. Fertig
```



Wer zeigt wann wohin?

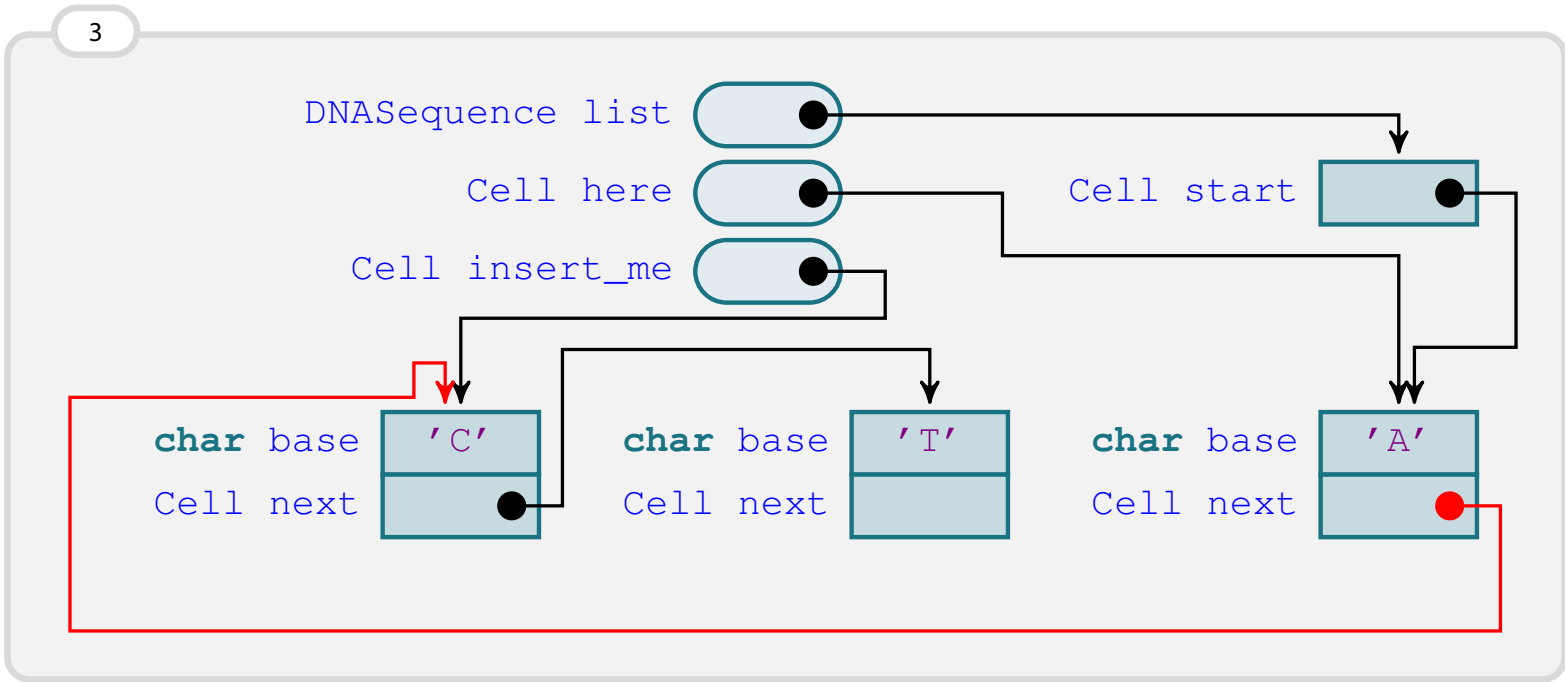

```
                                  // 1. Ausgangssituation
insert_me.next = here.next;       // 2. Erste Veränderung
here.next      = insert_me;       // 3. Fertig
```

2



Wer zeigt wann wohin?


```
                                  // 1. Ausgangssituation
insert_me.next = here.next;       // 2. Erste Veränderung
here.next      = insert_me;       // 3. Fertig
```



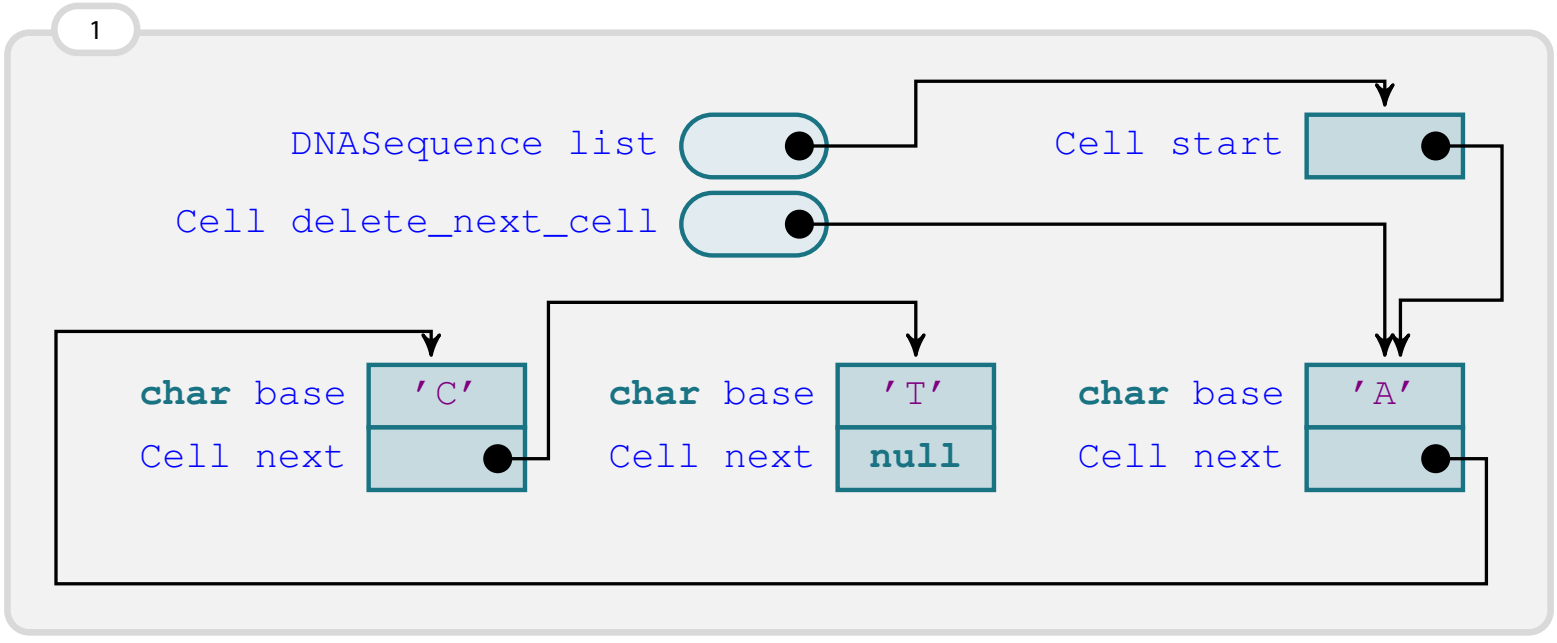
Wie löscht man ein Element aus der Mitte einer Liste?

- Wir wollen ein Element irgendwo in der Mitte einer Liste löschen.
- Dazu muss lediglich den *Nachfolger* des Vorgängers ändern.

```
class DNASequence {
  // ...
  void deleteAfter (Cell delete_next_cell) {
    // Löscht die Zelle, die auf delete_next_cell folgt:
    delete_next_cell.next = delete_next_cell.next.next;
  }
}
```

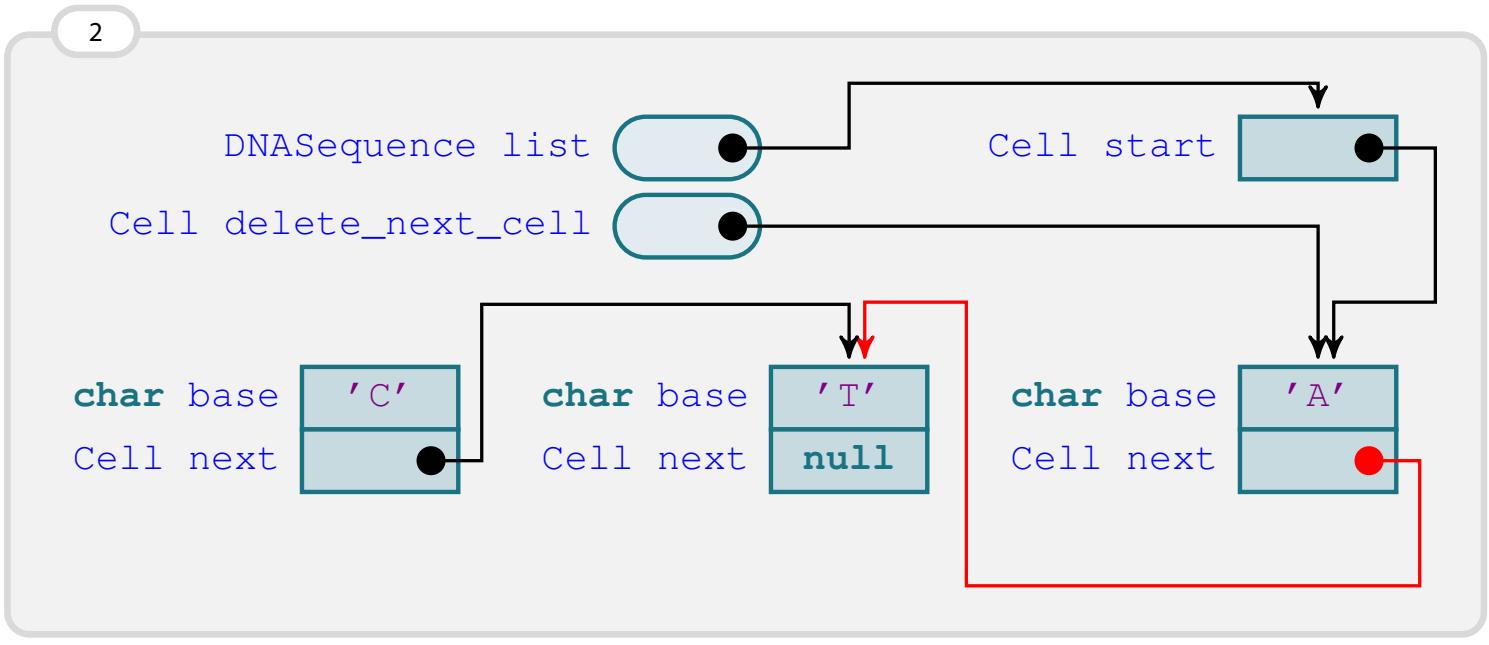
Wer zeigt wann wohin?


```
// 1. Vorher
delete_next_cell.next = delete_next_cell.next.next;
// 2. Nachher
```



Wer zeigt wann wohin?


```
// 1. Vorher
delete_next_cell.next = delete_next_cell.next.next;
// 2. Nachher
```



Wie verkettet man zwei Listen?

- Wir wollen aus zwei Listen eine Liste machen.
- Dazu muss der *Nachfolger* des letzten Elements der einen Liste der Anfang der zweiten Liste werden.
- Dazu muss man allerdings erst »das Ende finden« – dies macht man mit einer Iteration.

Zur Übung

Geben Sie den Code einer Methode zur Verkettung zweier Listen an:

```
class DNASequence {
  // ...
  void concatWith (DNASequence me) {
    // ?
  }
}
```

Zusammenfassung

Iteration über alle Elemente einer Liste

```
Cell cursor = list.start;
while (cursor != null) {
  tue_etwas (cursor);
  cursor = cursor.next;
}
```

Einfügen eines Elements in eine Liste

```
new_cell.next                = insert_after_this_cell.next;
insert_after_this_cell.next = new_cell;
```

Löschen eines Elements aus einer Liste

```
delete_cell_after_this_cell.next =
    delete_cell_after_this_cell.next.next;
```