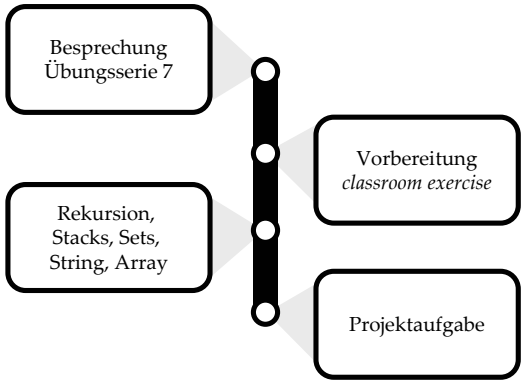
Montag, 5. & Dienstag, 7. Januar 2004



Shortest path for Santa

1. welche Stationen sind large?
   – *LINKED_LIST[METRO_STATION]*
2. welche Reihenfolgen (Permutationen) gibt es?
   – *LINKED_LIST[LINKED_LIST[METRO_STATION]]*
3. welche dieser Kombinationen ist die kürzeste?
   – *METRO_STATION*
4. markiere diese und animiere sie

1. Stationen, die *large* sind, suchen

```
create stations_to_visit.make
from map.city.metro_stations.start
until
map.city.metro_stations.after
loop
if map.city.metro_stations.item_for_iteration.segments_count > 8
then
stations_to_visit.extend(map.city.metro_stations.item_for_iteration)
end
map.city.metro_stations.forth
end
```

1. Stationen, die *large* sind, suchen

```
local
  stations_to_visit : LINKED_LIST[METRO_STATION]
...
create stations_to_visit.make
from
  map.city.metro_stations.start
until
  map.city.metro_stations.after
loop
  if map.city.metro_stations.item_for_iteration.segments_count > 8 then
    stations_to_visit.extend(map.city.metro_stations.item_for_iteration)
  end
  map.city.metro_stations.forth
end
```

2. Reihenfolgen (Permutationen) herausfinden

```
local
  permutations: LINKED_LIST[LINKED_LIST[METRO_STATION]]
...
permutations := permute(stations_to_visit)
```

3. Kürzeste aller Varianten finden

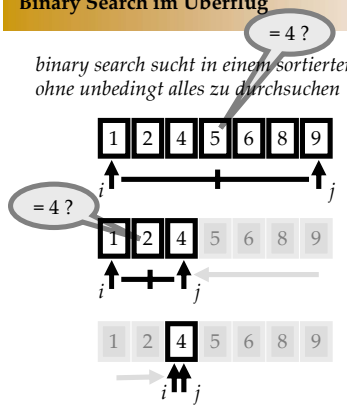
```
from
  permutations.start
until
  permutations.after
loop
  map.reset_route_selection
  map.route.wipe_out
  set_station_select (permutations.item)
  map.calculate_route
  if fastest = void or map.route.segments < hop_count then
    fastest := permutations.item
    hop_count := map.route.segments
  end
  permutations.forth
end
```

markiere alle Stationen, um die Länge danach zu berechnen

fastest ist am Anfang nicht definiert. nimm also erste Variante als bisher kürzeste

Binary Search im Überflug

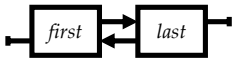
binary search sucht in einem sortierten Array ohne unbedingt alles zu durchsuchen

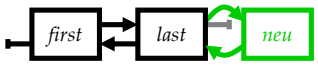



```
binary_search (a: ARRAY [INTEGER]; x : INTEGER): BOOLEAN is
    local
    i , j : INTEGER; m: INTEGER
    do
        from
            i := 1
            j := a.count
            Result := False
        until
            i > j or Result = True
        loop
            m := (i + j ) // 2
            if a.item (m) > x then
                    j := m − 1
            elseif a.item (m) < x then
                    i := m + 1
            else
                    Result := True
            end
        end
    end
```

Doppelt verkettete Zahlenliste

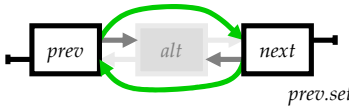
- eine Liste besteht aus keinem, einem oder mehreren Elementen
- man kann einfügen





last.set_next(neu)
neu.set_previous(last)
set_last(neu)

- und entfernen



prev.set_next(alt.next)
next.set_previous(alt.previous)

Vorbereitung *classroom exercise*

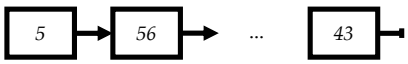
Themen:

- Rekursion
- Stacks
- Sets
- String
- Array
- Current

- ...und alles andere müsst ihr auch verstehen
  (Loop, Feature, Contracts)

Kurzübung Rekursion

z.B. finden in einer Liste



schreibe ein

```
feature
    finde( l : ELEMENT; w: INTEGER ) : BOOLEAN
```

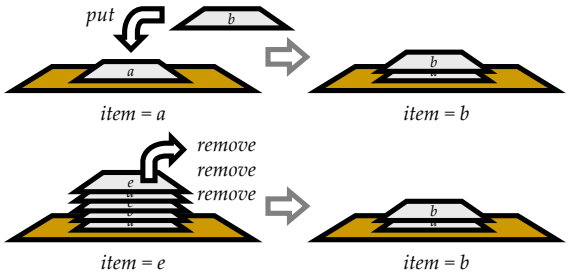
das **rekursiv** in der Liste *l* nach dem Wert *w* sucht.

Folgende Features sind bereits implementiert:

```
class ELEMENT
feature
    next : ELEMENT -- nächstes Element
    val: INTEGER -- Wert des Elementes
```

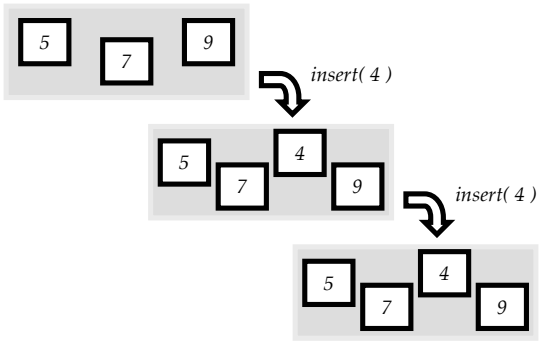
Stappel / Stack

auf einem Schreibtisch türmen sich Dokumente und man kann immer nur auf das oberste zugreifen



ein Set

ein Set ist eine Menge mit unterschiedlichen Elementen ohne Doubletten



Projektarbeit

Entertainment Information System

1. Modellieren
   - *überlegt euch, was für Klassen, Features, Locals, Loops ihr braucht*
   - *zeichnet dann ein Diagramm davon*
2. Implementieren
   - *schreibt als erstes das Diagramm in Klassen, Features etc. um*
   - *füllt erst dann die Features mit den nötigen Algorithmen*
3. Testen
   - *testet die drei auf dem Blatt gestellten Aufgaben*
   - *lasst euer Programm von einem anderen Team testen – im eigenen Programm findet man bald keine Fehler mehr, andere aber schon!*
4. Präsentieren
   - *präsentiert in 5 bis 10 Minuten euer Projekt – erklärt, was ihr beim Modellieren überlegt habt und welche Schwierigkeiten es gab*