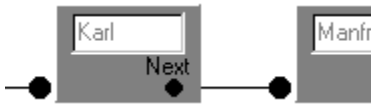
KURZANLEITUNG ZUM MODELLPROGRAMM (I)

Zur Datenorganisation allgemein:

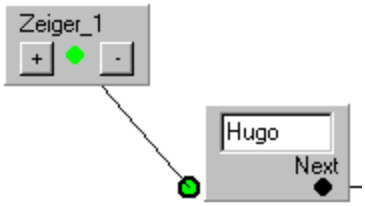
Die drei wesentlichen Merkmale der internen Verweistechnik lassen sich an den folgenden Beispielen festmachen:

1. Jedes Element besitzt eine Information, wo das nächste Element zu finden ist. Dadurch entsteht eine Verkettung der einzelnen Elemente:



Das Element mit dem Inhalt Karl hat einen Nachfolger mit dem Inhalt Manfred.
Außerdem gibt es ein Element, welches das Element mit dem Inhalt Karl als Nachfolger hat.

2. Das erste Element muß kenntlich gemacht werden, d. h. es muß einen Verweis geben, der auf das erste Element zeigt:



Das erste Element mit dem Inhalt Hugo ist durch den Zeiger „Zeiger_1“ kenntlich gemacht.

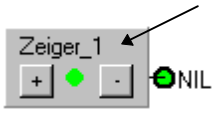
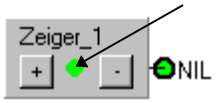
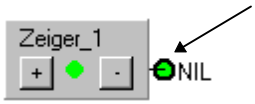
3. Das letzte Element muß deutlich machen, daß es keinen Nachfolger besitzt, d. h. es darf keinen undefinierten Nachfolger besitzen.

Das letzte Element mit dem Inhalt Manfred besitzt keinen Nachfolger.
Deshalb muß der Nachfolgerverweis auf ein definiertes Ende (NIL) gesetzt werden.
Ein undefiniertes Ende (?) ist nicht erlaubt.

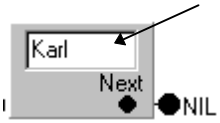
KURZANLEITUNG ZUM MODELLPROGRAMM (II)

Zur Bedienung des Programms:

1. Bedienung der Zeiger:

Zeiger erstellen	Drücken Sie auf den Button  .
Zeiger entfernen	Drücken Sie erneut auf den Button  .
Zeiger auf dem Bildschirm verschieben	Drücken Sie in den grauen Bereich des Zeigers und ziehen Sie den Zeiger auf eine andere Position. 
Neue Element am Zeiger erstellen	Drücken Sie den  -Button.
Element am Zeiger löschen	Drücken Sie den  -Button
Verweis eines Zeigers auf definiertes Ende setzen	Drücken Sie einmal auf den farbigen Punkt. 
Verweis eines Zeigers auf ein Element setzen	Drücken Sie in den umrandeten farbigen Punkt und ziehen Sie den an das Element, auf welches der Zeiger verweisen soll. 

2. Bedienung der Elemente:

Element auf dem Bildschirm verschieben	Drücken Sie in den grauen Bereich des Elements und ziehen Sie es auf eine andere Position. 
Nachfolger eines Elements auf definiertes Ende setzen	Drücken Sie auf den schwarzen Punkt unter „Next“. 
Nachfolger eines Elements auf ein anderes Element setzen	Drücken Sie den großen schwarzen Punkt und ziehen Sie diesen an das Element, welches Nachfolger werden soll. 
Ändern des Inhalts eines Elements	Drücken Sie einmal in das Textfeld und geben Sie den neuen Text ein. 

Grundsätzlich ist die Änderung eines Elemente nur dann möglich, wenn ein Zeiger darauf verweist (hellgrau dargestellt). Verweist kein Zeiger auf das Element, so ist es dunkelgrau dargestellt. Ausnahme: Ist ein Element durch keinen Zeiger mehr erreichbar, so wird es rot dargestellt (KARTEILEICHE!!!)

ARBEITSBLATT ZUM MODELLPROGRAMM

Führen Sie die beiden untenstehenden Aufgaben durch.
Sollten **Probleme** oder Merkwürdige Situationen auftauchen, so **notieren** Sie sich diese bitte in Stichpunkten (Situation vorher – Aktion – Situation nachher) in der dafür vorgesehenen Tabelle, damit wir zusammen Ihre Probleme im Anschluß klären können.

Aufgabe 1: Öffnen Sie die Datei VERKETT.ZGR aus Ihrem Verzeichnis und fügen Sie verschiedene neue Namen ein. Achten Sie dabei stets darauf, daß

1. die neuen Namen sortiert eingefügt werden
2. kein Datenmüll (rot unterlegtes Element) stehen bleibt. Verwenden Sie ggf. den Menüeintrag *Bearbeiten|Rückgängig* ( )
3. immer ein definiertes Ende der Verkettung vorliegt (NIL)
4. nicht unnötig viele Zeiger auf dem Bildschirm bleiben. Entfernen Sie diese mit einem erneuten Druck auf den zugehörigen Button.

Fügen Sie auch Namen ein, die am Ende (am Anfang) der Verkettung eingefügt werden müssen.
Speichern Sie Ihre Datei gelegentlich!

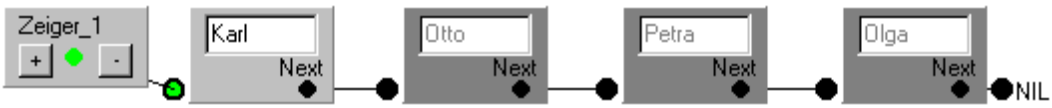
Aufgabe 2: Entfernen Sie nun aus der neuen Datei einige Namen. Achten Sie auch jetzt auf die vier oben genannten Punkte.
Entfernen Sie auch Namen, die am Ende (am Anfang) der Verkettung stehen.
Speichern Sie Ihre Datei gelegentlich!

Tragen Sie hier Ihre Problemsituationen ein:

Situation vorher (Skizze oder Beschreibung)	Aktion	Situation nachher (Skizze oder Beschreibung)

ARBEITSBLATT ZU DYNAMISCHEN LISTENSTRUKTUREN

Aufgabe 1: Im Wartezimmer eines Arztes sitzen die Personen Karl, Otto, Petra und Olga, die auch in dieser Reihenfolge vom Arzt empfangen werden sollen. Öffnen Sie die Datei WARTE.ZGR aus Ihrem Verzeichnis und schauen Sie sich die Verkettung an:



Simulieren Sie die folgenden Patientenbewegungen und notieren Sie sich die Aktionen, die wiederholt auftreten. Entwickeln sie ggf. Bedingungen an die Ausgangssituation, die eine Simulation der Patientenbewegungen vereinfachen würde:

a) Der Arzt empfängt den Patienten Karl. (Achten Sie darauf, dass die Komponente mit dem Namen Karl wirklich gelöscht wird und nicht nur als Karteileiche (rot unterlegt) stehen bleibt!)
b) Ein neuer Patient mit dem Namen Heinz betritt das Wartezimmer.
c) Der Arzt empfängt den Patienten Otto.
d) Der Arzt empfängt die Patientin Petra.
e) Eine neue Patientin mit dem Namen Tina betritt das Wartezimmer.

Speichern Sie zum Schluss die so entstandene Datei mit dem aktuellen Wartezimmer.

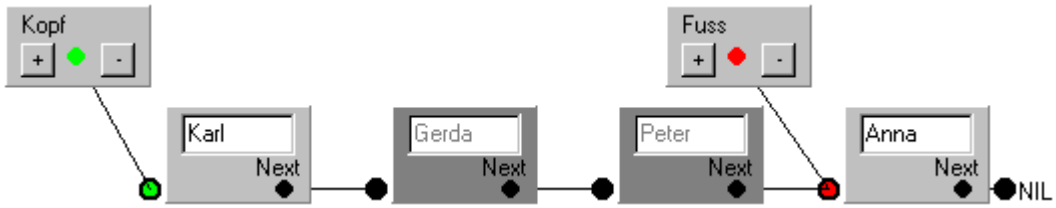
ARBEITSBLATT ZUR LISTENSTRUKTUR „SCHLANGE“

Aufgabe 1: Es ist morgens 8.00 Uhr früh, der Arzt ist noch nicht im Haus, die ersten Patienten kommen in die Praxis. Öffnen Sie die Datei WARTE2.ZGR mit dem Programm ADTMODEL.EXE – Sie finden die folgende Ausgangssituation vor:

Simulieren Sie nun die Einreihung der Patienten in eine Warteschlange und notieren Sie sich dabei jeden einzelnen Schritt, den Sie am Programm tätigen – **führen Sie also Protokoll**.

- Der Patient Karl kommt ins Wartezimmer.
- Die Patientin Gerda kommt ins Wartezimmer.
- Der Patient Peter kommt ins Wartezimmer.
- Die Patientin Anna kommt ins Wartezimmer.

Sind Sie fertig? Dann sollte die Schlange wie folgt aussehen:



Aufgabe 2: Entwickeln Sie nun mit Hilfe Ihrer Protokolle einen allgemeingültigen Algorithmus für das Anfügen eines Patienten an die Warteschlange. **Achten Sie auf eventuelle Spezialfälle**.

Aufgabe 3: Der Arzt kommt endlich in die Praxis uns ist bereit, Patienten zu empfangen. Schreiben Sie einen allgemeingültigen Algorithmus für das Löschen der Patienten aus der Warteschlange. Achten Sie auch jetzt auf eventuelle Spezialfälle!

ARBEITSBLATT ZUM BESUCHERZIMMER BEIM ARZT

– PRIVATPATIENTEN INKLUSIVE –

Aufgabe 1: Unser Arzt hält sich nicht mehr an die Vorschrift „Wer zuerst kommt malt zuerst“, sondern er bevorzugt Privatpatienten. Die Arzthelferin reiht die Privatpatienten nach ihrem Ermessen in die Wartereihenfolge ein. Zu Beginn ist das Wartezimmer mal wieder leer. Erstellen Sie dazu eine leere Schlange, indem Sie auf den Button  drücken. Ihre Aufgabe besteht nun darin, wie in der letzten Stunde die unten aufgeführten Patientenbewegungen zu simulieren und dabei Protokoll über Ihre getätigten Aktionen zu führen.

a) Der Kassenpatient Karl betritt das Wartezimmer
b) Die Kassenpatientin Tina betritt das Wartezimmer
c) Der Privatpatient Otto wird von der Arzthelferin vor den Kassenpatienten Karl gesetzt.
d) Der Arzt empfängt den Patienten Otto
e) Die Kassenpatientin Olga betritt das Wartezimmer
f) Der Arzt entdeckt im Wartezimmer die Bekannte seines Freundes: Tina. Er empfängt Tina als nächstes.
g) Der Privatpatient Peter wird von der Arzthelferin vor die Kassenpatientin Olga gesetzt.
h) Der Arzt hat eine Schwäche für Frauen. Er empfängt deshalb die Patientin Olga als nächstes.
i) Der Arzt empfängt den Patienten Karl
j) Der Arzt empfängt den Patienten Peter

ARBEITSBLATT ZUR DATENSTRUKTUR SCHLANGE IN DELPHI

Eine Typdeklaration für die Datenstruktur Schlange könnte in DELPHI wie folgt aussehen:

```
Type TElement = class
                  x,y:  integer;
                  Next: TElement;
                end;
     TQueue   = class
                  private
                  Kopf: TElement;
                  Fuss: TElement;
                  public
                  { und nun noch die Methoden }
                end;

Var  Queue:  TQueue;
```

Der im Unterricht entwickelt Pseudocode lässt sich in DELPHI durch folgende Operationen übersetzen:

Erstelle neuen Zeiger Hilf_1	`Var Hilf_1: TElement;`
Setze Zeiger Hilf_1 auf definiertes Ende	`Hilf_1 := NIL;`
Erstelle neues Element am Zeiger Hilf_1	`Hilf_1:= TElement.Create;`
Lösche Element am Zeiger Hilf_1	`Hilf_1.Free; { oder .Destroy }`
Belege Inhalt von Element am Zeiger Hilf_1	`Hilf_1.Inhalt := 'Inhalt';`
Setze Nachfolger von Element auf def. Ende	`Hilf_1.Next := NIL`
Setze Nachfolger von Element am Zeiger Hilf_1 auf Element am Zeiger Fuss (oder andere)	`Hilf_1.Next := Fuss;` (oder anderer Zeiger)
Setze Hilf_1 auf Nachfolger des Elements	`Hilf_1 := Hilf_1.Next;`
Lösche Zeiger Hilf_1	Lokale Zeiger werden automatisch entfernt!

Aufgabe 1: Nutze diese Gegenüberstellung um einen Algorithmus zu implementieren, der das Einfügen in die Datenstruktur Schlange beschreibt. Nutze dazu den folgenden, im Unterricht in ähnlicher Form entwickelten Pseudoalgorithmus:

Einfügen auch „enqueue“ genannt:

- Erzeuge Zeiger Hilf
- Erstelle neues Element am Zeiger Hilf
- Belege Inhalt des Elements am Zeiger Hilf
- Setze Nachfolger des Elements am Zeiger Hilf auf NIL
- wenn Kopf auf NIL zeigt *{d. h. die Schlange ist leer }*
  dann: • Verbinde Zeiger Kopf mit dem Element am Zeiger Hilf
  sonst: • Setze Nachfolger vom Element am Zeiger Fuss auf El. an Zeiger Hilf
- Verbinde Zeiger Fuss mit Element am Zeiger Hilf
- Lösche Zeiger Hilf

Aufgabe 2: Implementiere einen Algorithmus, der das Löschen aus der Datenstruktur Schlange beschreibt.

DIE DATENSTRUKTUR SCHLANGE IN DELPHI

Deklaration:

Modellprogramm	DELPHI
Druck auf den  –Button	**Type** TElement = **class** x,y: integer; Next: TElement; **end;** TQueue = **class** **private** Kopf: TElement; Fuss: TElement; **public** *{ und nun noch die Methoden }* **end;** **Var** Queue: TQueue;

Notwendiger Befehlssatz:

Modellprogramm	DELPHI
Erstelle neuen Zeiger Hilf_1	**Var** Hilf_1: TElement;
Setze Zeiger Hilf_1 auf definiertes Ende	Hilf_1 := NIL;
Erstelle neues Element am Zeiger Hilf_1	Hilf_1:= TElement.Create;
Lösche Element am Zeiger Hilf_1	Hilf_1.Free; *{ oder .Destroy }*
Belege Inhalt von Element am Zeiger Hilf_1	Hilf_1.Inhalt := 'Inhalt';
Setze Nachfolger von Element auf def. Ende	Hilf_1.Next := **NIL**
Setze Nachfolger von Element am Zeiger Hilf_1 auf Element am Zeiger Fuss (oder andere)	Hilf_1.Next := Fuss; (oder anderer Zeiger)
Setze Hilf_1 auf Nachfolger des Elements	Hilf_1 := Hilf_1.Next;
Lösche Zeiger Hilf_1	Lokale Zeiger werden automatisch entfernt!

Beispiel: Einfügen in eine Schlange:

Oberflächen-Unit:

```
procedure TForm1.B_EnqueueClick(Sender: TObject);
var e: TElement;                          { Zeiger "Hilf" erstellen          }
begin
  e:= TElement.Create;                    { Erstelle neues El. am Zeiger Hilf }
                                          {  hier wird Nachfolger auf NIL gesetzt }
  e.x:= SpE_1.Value;                      { Belege Inhalt des El. an Hilf     }
  e.y:= SpE_2.Value;
  queue.enqueue(e);                       { Jetzt einfügen in Schlange...     }
end;
```

Unit U_Queue:

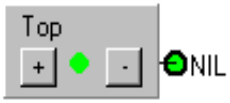
```
procedure TQueue.enqueue(e: TElement);    { Neues Element in Schlange         }
begin
  if empty                                { Wenn Kopf auf NIL zeigt (Hilfroutine) }
    then begin
           Fuss:= e;                      { dann El. als erstes einreihen     }
           Kopf:= e;                      { und Fuss auf neues Element setzen }
         end
    else begin
           Fuss.Next:= e;                 { sonst Element hinten eingliedern  }
           Fuss:= e;                      { Fuss-Zeiger weiterrücken          }
         end;
end;
```

TEST ZU DYNAMISCHEN LISTENSTRUKTUREN

Der bereits bekannte Arzt war früher einmal Beamter und hat seine Gewohnheiten aus damaliger Zeit übernommen: Kommt ein neuer Patient in das Wartezimmer, so wird seine Akte immer oben auf einen Stapel gelegt. Empfängt der Arzt einen Patienten, so nimmt er sich die oberste Akte vom Stapel herunter und bittet den entsprechenden Patienten zur Untersuchung. Eine Besucherreihenfolge könnte wie folgt aussehen:

Patientenbewegung:	**Aktenstapel:**
Otto kommt ins Wartezimmer	Otto
Karla kommt ins Wartezimmer	Karla Otto
Peter kommt ins Wartezimmer	Peter Karla Otto
Der Arzt empfängt Peter	Karla Otto
Olga kommt ins Wartezimmer	Olga Karla Otto
Der Arzt empfängt Olga	Karla Otto
Der Arzt empfängt Karla	Otto
Der Arzt empfängt Otto	_____

Aufgabe 1: Schreiben Sie einen verbalen Algorithmus, der das Ablegen einer neuen Patientenakte auf den Aktenstapel beschreibt. Die Ausgangssituation ist durch die folgende gegeben:



Dabei verweist der Zeiger Top zu jeder Zeit auf das erste Element (oberste Akte auf dem Stapel).

Aufgabe 2: Schreiben Sie einen verbalen Algorithmus, der das Löschen aus dieser Datenstruktur beschreibt. Achten Sie darauf, daß der Arzt stets den Patienten der obersten Akte empfängt.

Hinweis: **Berücksichtigen Sie sämtliche Spezialfälle!!!**