# Tag

## Einführung

* Sinnhaftigkeit von dynamischen Datenstrukturen => **flexible** Nutzung von Speicher, Datenobjekte können während der Laufzeit erzeugt und wieder freigegeben werden.

## Zeiger

Tafel: Zeiger erklären (siehe Folie 2) oder Beispiel von Unterlagen (Blatt, siehe Ordner) mit INTEGER\_REF

* Erzeugung und Zuweisung eines Zielobjektes an das Zugriffsobjekt
  + Auswertung der Untertypangaben bzw. des Ausdrucks
  + Erzeugung des dyn. Zielobjekts
  + ggf. Initialisierung des dyn. Zielobjekts
  + Rückgabe des Verweises (Adresse) des Zielobjekts an das Zugriffsobjekt (Zeiger)

### Syntax

* Zeiger-Datentyp anlegen (Stack)
* Dynamisches Datenobjekt anlegen (Heap)

### Eigenschaften

* **Zugriffobjekte** (statisch) sind typgebunden
* Zugriffobjekte nur auf dynamisch erzeugte **Datenobjekte** anwendbar!? (vorerst)
* Zugriffswerte des Zeigers referenzieren immer ein Datenobjekt oder kein Datenobjekt (**null**-Pointer)
* dynamsiche **Datenobjekte** sind **i m m e r** beschränkt!
* Null-Pointer-Zugriffe führen zum Constraint\_Error
* Operationen auf Zeiger (Zugriffsobjekte) sind
  + Deferenzierung (.all),
  + Vergleiche (=,/=),
  + Zuweisung (:=) und
  + Allokation (new )

### Anwendung von Zeiger mit skalaren Datentypen

* Template: ***Template\_Pointer*** ( nur skalare Datentypen)
* Foliensatz bis Seite 16
* Übung: Aufgabe 1

### Anwendung von Zeiger mit strukturierten Datentypen

* Template: ***Template\_Pointer*** ( Rest vom Template)
* ***Template\_String***
* Übung: ***Uebung2NEU*** im Vorgabeordner von LT machen lassen => in Übungen integriert
* Übung: Aufgabe 3a) und
  + 1. Schritt: nur einen String-Zeiger anlegen und dem einen Namen zuweisen, der über Tastatur eingegeben wurde und diesen wieder ausgeben
  + 2. Schritt: Array auf String-Zeiger auf Stack anlegen (feste Größe) und dieses befüllen und anschließend ausgeben.
* Aufgabe 3b)
  + Anzahl Teilnehmer abfragen
  + dynamisch Array für die Teilnehmer anlegen
  + Einlesen der Namen, im dyn. Speicherbereich ablegen und im Array verwalten
  + Ausgabe der Namen
* Foliensatz bis Seite 23 (optional)

### Freigabe von dynamischen Datenobjekten

* Foliensatz bis Seite 27 (optional) , Tafelanschrift oder Skript
* Template: ***Template\_Freigabe***
* Aufgabe 2c)
  + Löschen/Freigabe der String-Objekte
  + Löschen/Freigabe des Arrays

### SWPÄ

* Rechnung-Programm: Kapitel 3.10 Dynamische Datenstrukturen anwenden

# Tag

## Wiederholung

* Tafel: Zeiger auf Reihung mit INTEGER-Werten von LT
* Erweiterung (hoersaal\_Ptr) : Hoersaal-Liste soll in einer dyn. Reihung (Heap) verwalten werden (***hoersaal\_Ptr\_Dyn***)
  + Es soll über eine Abfrage, soll eine weitere Person aufgenommen werden, die Reihung dynamisch erweitern.
  + Die Person soll vorne bzw. hinten angefügt werden (Modus soll vorab abgefragt werden).
  + Nach Abschluss der Eingabe sollen alle Werte ausgegeben werden. Dazu schreiben Sie eine Prozedur, die diese Aufgabe für die erledigt.
* Problematik : Hinführung zu Listen (Notwendigkeit wecken)

## Listen

### Einführung

* Tafel : graphische Einführung, Prinzip von Listen erklären, Listenelement, rekursive Struktur

### Syntax

* Listenelement-Definition: Unvollständige Typdeklaration, Zeigertyp, Vollständige Typdeklaration (Ergänzung) .
* Template: ***template\_Listen***

### Einfügen in eine Liste

* Leere Liste
* vorne Einfügen in einer Liste
* hinten Einfügen in einer Liste

### Durchlaufen einer Liste

### Übungen

* Aufgabe: 4 und
* Aufgabe: 5
  + 1 Paket schreiben
    - 1.1 Spezifikation schreiben Listenpaket\_Ado\_Gen (globale\_daten\glob\_dynDatenStruk)
      * 1.1.1 Element als **private**
      * 1.1.2 Generische Schleife
      * 1.1.3 Elemente in einer Liste verwalten
    - 1.2 Body schreiben
  + 2 Hauptprogramm ***main\_hoersaal\_liste***
    - 2.1 Paket einbinden
    - 2.2 generisches Paket auf STRING\_REF ausprägen
    - 2.3 Prozedur für Anzahl\_Teilnehmer ermitteln
    - 2.4 Prozedur für Ausgabe der Namen erstellen
    - 2.5 generische Schleifen ausprägen
    - 2.6 Abfrage-Schleife erstellen, die mit Beenden verlassen wird
      * 2.7 Namen abfragen und im Paket hinterlegen (hinten einfügen)
      * 2.8 Weitere Namen JA/Nein
    - 2.9 Ausgabe Anzahl Teilnehmer und Ausgabe der Namen über generische Schleife

# Tag

## Wiederholung

* Zusatzaufgabe eine Liste kopieren (Tiefe Kopie) Funktion schreiben (Tiefe Kopie)
* Besprechung Aufgabe 5

bis 9:00 Uhr

## Listen – Fortsetzung –

Siehe Template: ***template\_Listen***

### Einfügen in der Mitte der Liste (als SWPÄ-Analyse an LT)

* Einfügestelle suchen -> Weiterschalten mit nachschleppen
* Einfügen des Elementes

### Element löschen aus einer Liste (als SWPÄ-Analyse an LT)

* Das zu löschende Element suchen -> Weiterschalten mit nachschleppen
* Das zu löschende Element ausketten
* Evtl. Foliensatz bis Seite 52

### Wiederholtes Löschen in einer Liste (Elemente mit dem gleichen Schlüssel)

* Skript: Seite 26

### Rekursive Vorgehensweise (Skript)

* Evtl. aus Teil 3 jetzt erst Rekursion erklären
* Durchlaufen einer Liste (Seite 27)
* Einfügen eines Elements in eine Liste (Seite 27)  
   (vielleicht als rekursive Funktion ausführen, leichter verständlich)
* Entfernen eines Elementes aus einer Liste (Seite 28)

### Ergänzung rekursive Algorithmen

* HP: ***main\_listenpaket***
* globale\_Daten\glob\_DynDatenStruk\***listenpaket\_adt*** (Spezifikation vorgeben)

## Übungen

* Aufgaben: 6-9
  + Flavius-Methode (Aufgabe 6) mit Paket: Einfache Listenverwaltung (Aufgabe 5) lösen (Erweiterung)
    - Lösch-Methoden integrieren für Verwaltungselement  
      Hinweis: Löschen des Inhaltes wird vom Nutzer übernommen!
    - Ohne Ringschluss, aber automatisch wieder Liste von vorne beginnen lassen (auslesen)
    - Nutzer ist für das Löschen des Namens verantwortlich. Zusätzlichen Parameter, welche Person (zweite, dritte,..) ausgefügt werden soll. Flag für Gewinner (Letzte Person in der Liste), Rückgabe der Person (STRING\_REF)

# Tag

## Wiederholung

* Listen, Aufgaben besprechen

bis 9:00 Uhr

## Bäume

### Definitionen (Tafel erklären)

* nur ein Knoten hat **k e i n e n** Vorgänger => Wurzel
* jeder weiterer Knoten hat genau **e i n e n** Vorgänger
* jeder Knoten ist von der Wurzel aus nur auf genau **e i n e m** Weg erreichbar
* Was ist ein Baum? -> Knoten mit Beziehungen (Beschränkungen) Binärbaum
  + Wurzel
  + Ast
  + Tiefe
  + Teilbaum
  + Blatt
  + leerer Baum

### Geordneter Binärbaum

* Sortiert abgelegt links kleiner rechts größeres Element (Schlüssel)
* Tafelbild erstellen

### Operationen auf Binärbäume

Rekursiv: An der Tafel erarbeiten

* Template: ***Template\_baeume*** , umfasst Grundlagen und einigen Übungen und Zusatzaufgaben

#### Einfügen im Binärbaum

* Es wird immer ein Blatt eingefügt.

#### Durchlaufen

**Wann** wird die ***Aktion*** ausgeführt? Besuchen der Knoten im Baum

* In-Order mit LT durchspielen
* Pre-Order
* Post-Order

Übungen: Aufgaben 10 und 11 in ***Template\_baeume***

#### Löschen im Binärbaum

* hat keinen Nachfolger
* hat einen Nachfolger
* hat zwei Nachfolger

### Übungen/Wiederholung

* Aufgabe 12 (suchen im Binärbaum)
* Weitere Übungen zur Wiederholung aus ***Template\_baeume***:
  + 20 Zufallswerte zwischen 1 - 1000 sortiert in Baum einfügen  
    (keine doppelten Werte aufnehmen)
  + Ausgabe der Baumstruktur (Ausgabe\_Info - Pre-Order-Durchlauf)
  + kleinsten und größten Wert aus Baum ermitteln
  + gesamten Baum löschen (Post-Order-Durchlauf)  
    -- nur Blätter werden **gelöscht** –
* ***evtl. Baumverwaltung\_adt***, ***main\_Baumverwaltung*** oder
* ***Baumpaket\_Ado\_Gen***, ***main\_baumpaket\_ado***
* Evtl. als Zusammenfassung Foliensatz bis Seite 69

# Tag

## Wiederholung

* Übungen nachbesprechen

bis 9:00 Uhr

## Gerichteter Graph

### Definition

* Knoten sind die Ecken eines Graphen.
* Kanten sind die Beziehungen zwischen zwei Knoten (Knotenpaar)
* Verweist eine Kante auf sich selbst, dann spricht man von einer **Schlinge**
* Knoten werden durch Punkte/Kreise und Kanten mit Linien mit Pfeil zwischen Knoten darstellt

### Definition Symmetrischer Graph

Gibt es zu jeder Kante eines gerichteten Graphen auch eine entgegengesetzte Kante (inverse) im Graph, dann spricht man vom **symmetrischen** Graphen.

### Aufbau eines Graphen (Tafel)

* Beispiel: Wegenetz siehe Skizze gerichteter Graph und
* Template: ***Template\_Graphen*** oder
* HP: ***graphnutzer*** und   
  Paket: Globe\_Daten\Glob\_DynDatenStruk\***graphverwaltung***   
  (Ausfügen von Kante und Knoten)

#### Einfügen von Knoten

#### Einfügen von Kanten

Die beiden Einfüge-Prozeduren an der Tafel gemeinsam erarbeiten und danach von LT programmieren lassen.

#### Durchlaufen aller Knoten und Kanten

* Ausgabe\_Wegenetz Vorgabe von **n u r** dieser Prozedur und die Ausgabe
  + => alle anderen Prozeduren von LT schreiben lassen

#### Ausfügen einer Kante

Anhand des Skripts erklären oder erklären lassen (mit Tafelbild, graphisch die Schritte erläutern)

#### Ausfügen eines Knotens

Die beiden Ausfüge-Prozeduren von LT erstellen lassen, dabei aber die einzelnen Schritte an der Tafel anschreiben. (siehe Graphenverwaltung –Paket in Globale\_Daten\glob\_DynDatenStruk)

* Problem in Teilschritte zerlegen: (mögliche Vorgehensweise)
  + Knoten suchen
  + Prozedur zum Entfernen aller Kanten eines Knotens schreiben
  + Über alle Knoten laufen und ausfügen einer Kante (siehe 5.2.3.4) nutzen (Start\_Knoten-aktueller KnotenNr, ZielknotenNr ist der zu entfernende Knoten)
  + Prozedur zum Entfernen eines Knotens schreiben
* Nach jedem Teilschritt die Ausgabe-Prozedur (siehe 5.2.3.3) ausführen, ob das erwartete Ergebnis eingetroffen ist (Test)

### Zusammenfassung

* Evtl. Foliensatz bis Seite 78

### Übungen

* Aufgabe 13 -16

## Erweiterung Zeiger

* Template: ***template-advanced\_ptr***
* Evtl. Foliensatz bis Seite 89

# ToDo

* evtl. Lösungen von Graphen-Aufgaben 13-16 anpassen!
* Analyse-Aufgabe (SWPÄ) mit Zeigern erstellen.
  + (Logische Operatoren: or, and, xor), Anwesenheit von Personen (TRUE/FALSE) mit Ergebnis und Wetter verknüpfen oder …
  + Hauptprogramm vorgeben und Teile in Paket auslagern
* Zugriffe auf Array's Beispiele suchen aus evtl. HD - F-4F und
* auch für Wiederholungsaufgaben und für Test erstellen