

# Arbeit mit Pointern - einfach verkettete Listen

## Sonderübung

Nadine Schärmann & Thomas Klütz

PROG

27. April 2020

1. Organisatorisches
2. Die Einkaufsliste-Standardfunktionen
3. Wiederholung: Gefahren beim Umgang mit Pointern
4. Noch mehr Funktionalität für die Einkaufsliste
5. Warteschlange

# Organisatorisches

- Blatt9 bis 08.05.2020, Abgabe im Abgabebriefkasten im OPAL
- Fragen zur Übung im Matrix-Chatraum
- Lösungen der letzten Sonderübung im gitlab:  
<https://gitlab.mn.tu-dresden.de/wwalter/PROG-material-public/>
- Videos der letzten sonderübungen findet ihr her:  
<https://gitlab.mn.tu-dresden.de/s9210296/prog-videos>

Gebt uns gerne Bescheid, wenn ihr Wünsche, Vorschläge o. Ä. bzgl. der Sonderübung habt.

# Einkaufsliste

Was haben wir in den letzten zwei Woche schon getan?

- Geeignete Datentypen erstellt
- Aufbau der Liste bei bekannter Länge
- Ausgabe der Liste
- Arbeit mit Liste unbekannter Länge
- Löschen der gesamten Liste von vorne

Was wollen wir heute machen?

- Aufbau einer alphabetisch sortierten Liste
- ZUSATZ: Finden und Löschen eines Elements

# Wiederholung: Gefahren beim Umgang mit Pointern

- Verwendung/ Dereferenzierung/ Abfrage des Zustands eines Pointers in einem undefinierten Zustand
- dangling pointer: Zugriff auf bereits gelöscht Objekt (kann unter anderem passieren, wenn mehrere Pointer auf das selbe Objekt zeigen und einer davon deallokiert wird)
- Speicherleichen (memory leaks): Wir lassen Objekte im Speicher zurück, die wir nicht mehr erreichen können

Der Compiler findet diese Fehler im Normalfall nicht!

**Ausweg:** Das Programm **valgrind**

`gfortran -g Dateiname -o Programmname`

`valgrind .\ Programmname`

# Noch mehr Funktionalität für die Einkaufsliste

Wir schreiben eine alphabetisch sortierte Einkaufsliste. Dafür brauchen wir die folgenden Subroutinen:

## **Einfügen eines Elements in eine alphabetisch sortierte Liste**

Schreibe eine Subroutine, in welcher ein Artikel an die richtige Position innerhalb einer sortierten Liste eingefügt wird. Die Subroutine bekommt den Head-Pointer der Liste sowie den Namen und die Menge des einzufügenden Artikels übergeben. Innerhalb der Subroutine wird iterativ der Artikelname jedes Listenelements mit dem Namen des einzufügenden Artikels verglichen. Denke auch an die Sonderfälle, dass der Artikel an erster oder letzter Position eingefügt werden muss oder die Liste leer ist.

Beachte, dass man in einer einfach verketteten Liste nur Elemente nach einem anderen Element einfügen kann (abgesehen von der ersten Position).

**ZUSATZ:** Falls das Produkt schon auf der Liste steht, soll nur seine Anzahl erhöht werden und es wird nicht noch einmal hinzugefügt.

**ZUSATZ:** Schreibe eine neue Subroutine zum Erstellen der Liste. Darin soll zunächst der erste Artikel eingelesen und auf die Liste geschrieben werden. Anschließend ist jeder weitere eingelesene Artikel mithilfe obiger Subroutine in die Liste einzufügen.

## ZUSATZ: Finden und Löschen eines bestimmten Elements

Schreibe eine Funktion, die als Parameter den Namen eines Artikels übergeben bekommt. Dieses wird dann in der Liste gesucht. Falls es vorhanden ist, soll es gelöscht werden und `.TRUE.` soll zurückgegeben werden. Andernfalls gibt die Funktion `.FALSE.` zurück.

Hinweis: Beachte, dass es leichter ist, den Nachfolger eines Elements zu löschen als das Objekt selbst.

## ZUSATZ: Warteschlange

Unsere Einkaufsliste kann auch als Warteschlange verstanden werden: Eine Warteschlange ist eine Liste, bei der Elemente ausschließlich am Tail eingefügt und nur am Head gelöscht werden.

Füge zur Typdefinition der Liste einen Tail-Pointer hinzu. Schreibe nun eine Subroutine zum Anfügen eines Elements am Listenende, welche den Tail-Pointer einer Liste sowie den Namen und die Menge des anzufügenden Artikels übergeben bekommt. Außerdem soll eine Subroutine zum Löschen eines einzelnen Elements am Listenanfang geschrieben werden.

Beachte die Sonderfälle (z. B. Löschen aus einer leeren Liste, Nachfolger des übergebenen Tail-Pointers ist assoziiert, ... )