

# Arbeit mit Pointern - einfach verkettete Listen

## Sonderübung

Nadine Schärmann & Thomas Klütz

PROG

21. April 2020

1. Organisatorisches
2. Die Einkaufsliste-Standardfunktionen
3. Gefahren beim Umgang mit Pointern
4. Noch mehr Funktionen für die Einkaufsliste

# Organisatorisches

- Blatt8 bis 21.04.2020, Abgabe im Abgabebriefkasten im OPAL
- Fragen zur Übung im Matrix-Chatraum
- Lösungen der letzten Sonderübung im gitlab  
(<https://gitlab.mn.tu-dresden.de/s9210296/PROG-material-public/-/tree/master/Sonderuebung>)

Gebt uns gerne ein Feedback darüber, ob ihr die Aufgaben bearbeitet und gelöst habt, wo es dabei vielleicht Schwierigkeiten gab, ob die Tonqualität des ersten Videos gut war oder ob ihr euch z.B. ein live-Zoom-Meeting wünscht.

# Einkaufsliste

Was haben wir letzte Woche schon getan?

- Geeignete Datentypen erstellt
- Aufbau der Liste bei bekannter Länge
- Ausgabe der Liste

Was wollen wir heute machen?

- Arbeit mit Liste unbekannter Länge
- Löschen der gesamten Liste von vorne
- ZUSATZ: Finden und Löschen eines Elements

# Einkaufsliste

Benutzt euer Programm der letzten Sonderübung, sollte euer Programm nicht funktionieren, könnt ihr dieses mit dem Lösungsvorschlag im gitlab vergleichen.

## **(Wiederholung) Ausgabe der Liste**

Wir befinden uns nun im Supermarkt und arbeiten die Liste der Reihe nach ab. Schreibe dafür eine Subroutine: In einer Schleife wird in jeder Iteration ein weiteres Produkt in den Einkaufswagen gelegt. Der Name und die Menge sind auszugeben.

## **Beliebige Länge**

Nun soll die Länge der Liste beliebig sein. Beim Einlesen soll zu Beginn jedes Schleifendurchlaufs abgefragt werden, ob die Liste der Einkäufe noch weitergeht. Falls nicht, ist die Schleife zu verlassen. Auch das Abarbeiten der Liste im Supermarkt muss nun angepasst werden. In jedem Schleifendurchlauf muss überprüft werden, ob wir das Ende der Liste erreicht haben, d. h. ob der Nachfolger-Pointer nicht assoziiert ist.

## Löschen der gesamten Liste

Zu Hause wird ausgepackt! Schreibe eine Subroutine, in welcher iterativ (von vorn beginnend) ein Produkt aus dem Beutel genommen und im Schrank verstaut wird. Ein entnommenes Produkt ist aus der Liste zu löschen.

## ZUSATZ: Finden und Löschen eines bestimmten Elements

Schreibe eine Funktion, die als Parameter den Namen eines Lebensmittels 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.

# Gefahren beim Umgang mit Pointern

- Verwendung/ Dereferenzierung / Abfrage des Zustands eines Pointers in einem undefinierten Zustand
- dangling pointers: Zugriff auf bereits gelöschttes Objekt (kann unter anderem passieren, wenn mehrere Pointer auf dasselbe 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`

# Mehr Funktionen für die Einkaufsliste

Jetzt sollen die Lebensmittel in alphabetischer Reihenfolge aufgeschrieben werden.

Schreibe eine Funktion *kleiner*, die als Eingabe zwei CHARACTER bekommt. Sie soll .TRUE. zurückgeben, falls das erste Wort in alphabetischer Reihenfolge vor dem anderen kommt, .FALSE. sonst.

Schreibe eine Subroutine, die den Head-Pointer der Liste und Name und Anzahl des einzufügenden Objekts übergeben bekommt. Füge dieses Element nun an der richtigen Stelle ein.

Beachte, dass man in einer einfach verketteten Liste nur Elemente nach einem anderen Element einfügen kann.

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