Matrikel: 306953 Name: Hauke Buder

# Einsendeaufgabe 2

## Aufgabe 1 Gleitkommazahlen

Im Standard IEEE754 für Gleitkommazahlen können nicht nur „normale“ Zahlen dargestellt werden, sondern auch spezielle Zahlen (wie NaN, +Inf, und –Inf) repräsentiert werden. Geben Sie die Binärdarstellung der folgenden Zahlen in IEEE754 Half Precision (1.5.10-Gleitkommazahl) an:

a) 0,0

*0 00000 0000000000 = 0,0*

b) +inf (positiv Unendlich)

*0 11111 0000000000 = positiv unendlich*

c) NaN (not a number)

*0 11111 1111111111 = NaN*

Dabei hilft Ihnen möglicherweise die Quelle https://en.wikipedia.org/wiki/Half-precision_floatingpoint_format.

## Aufgabe 2 Datenstrukturen

Formulieren Sie das Löschen eines beliebigen Listenknotens aus einer einfach verketteten Liste *L* als Algorithmus. Ein Listenknoten besteht dabei aus dem Attribut *data* für die Daten des Knotens und dem Attribut *next* für die Referenz/ Zeiger auf den nächsten Listenknoten.

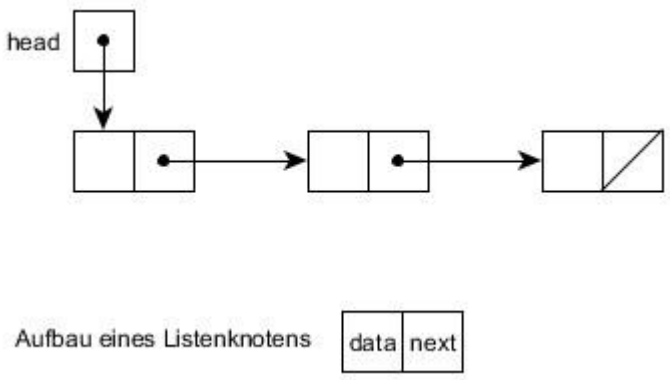


*Abbildung 1 Schema einer einfach verketteten Liste.*

Gehen Sie am besten nach dem Teile-und-Herrsche-Prinzip vor und teilen Sie das Problem in drei leichter zu lösende Unterprobleme ein:

- Löschen des ersten Listenknotens
  *$ziel = $head.next*

  *$head.next = $ziel.next*

  *Setze die Variable „ziel“ auf das nächste Listenelement und schreibe den Wert „Next“ des nächsten Listenelements in den „Next“ Wert von „head“*

- Löschen des letzten Listenknotens

*$aktuellesElement = $head*

*While ($aktuellesElement.next -ne NULL){*

*$nachfolgerLoeschen = $aktuellesElement*

*$aktuellesElement = $aktuellesElement.next*

*}*

*$Nachfolgerloeschen.next = NULL*

*Setze die Variable „aktuellesElement“ auf „head“ Während der „next“ Wert des aktuellen Elements nicht NULL ist schreibe erst das „aktuelleElement“ in „nachfolgerLoeschen“ und inkrementiere aktuellesElement auf Element.next.*

*Überschreibe „Nachfolgerlöschen.next“ mit NULL*

- Löschen eines mittleren Listenknotens

*$zuloeschen = [INHALT]*

*$zwischenablage =$head*

*foreach ($element in $liste){*

*If ($element.data -eq $zuloeschen){*

*$liste[$zwischenablage].next = $element.next*

*}*

*$zwischenablage = $element*

*}*

*Schreibe den Inhalt, der zum Löschen gesucht wird, in die Variable „zuLoeschen“.*

*Gehe Element für Element durch die Liste. Wenn der Inhalt mit dem aktuellen Element übereinstimmt so schreibe den „next“ Wert des Elements in den „next“-Wert des in der Variable„Zwischenablage“ gespeicherten Eintrags.*

*Schreibe das aktuelle Element in die Variable „Zwischenablage“ für den nächsten Schleifendurchlauf*

*Die Lösung zum Löschen eines mittleren Listenknotens funktioniert gleichzeitig auch für den ersten bzw. letzten Listenknoten, daher ist eine Zusammenfassung nicht notwendig.*

*Als Eingabe muss der Inhalt des zu löschenden Elements übergeben werden sowie die zu verarbeitende Liste.*

*Das Löschen erfolgt durch Überschreiben des „next“ Wertes des Vorgängers mit dem „next“ Wert des Nachfolgers*

Lösen Sie die drei Unterprobleme einzeln und setzen Sie die Teillösungen zu einer gesamten Lösung zusammen.

Verwenden Sie eine für Sie sinnvolle Darstellung des Algorithmus, also textuelle Beschreibung, Pseudocode, Flussdiagramm oder eine Mischung aus den zuvor genannten Darstellungen und benennen Sie explizit, was die Eingabedaten und Ausgabedaten Ihres Algorithmus sind.

Matrikel: 306953 Name: Hauke Buder

# Aufgabe 3 Algorithmen

Gegeben ist die Liste L=[3, 2, 0, 1, 4] mit der Länge |L|=5.

Wenden Sie den Algorithmus in Abbildung 2 auf die Liste an und bestimmen Sie dabei die Anzahl der Vergleiche und die Anzahl der Tauschoperationen (hier swap genannt).

*Es werden 29 Vergleiche, davon 10 mit Bezug auf die Werte, und 5 Swaps ausgeführt.*

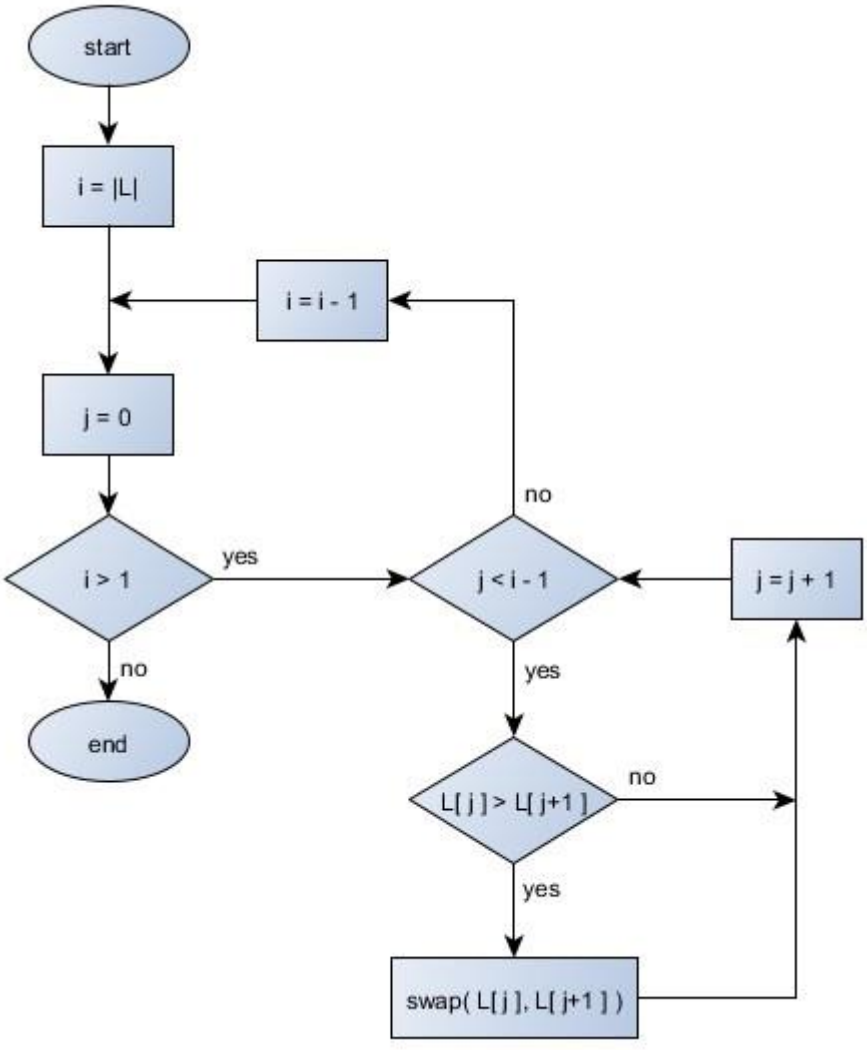


*Abbildung 2 Flussdiagramm einer Bubble-Sort Implementierung.*

5 größer 1 -> j

0 kleiner 4 -> j

3 größer 2 -> j -> 2,3,0,1,4

1 kleiner 4 -> j

3 größer 0 -> j -> 2,0,3,1,4

2 kleiner 4 -> j

3 größer 1 -> j -> 2,0,1,3,4

3 kleiner 4 -> j

3 größer 4 -> n

4 kleiner 4 -> n

4 größer 1 -> j

0 kleiner 3 -> j

2 größer 0 -> j -> 0,2,1,3,4

1 kleiner 3 -> j

2 kleiner 1 -> j -> 0,1,2,3,4

2 kleiner 3 -> j

2 größer 3 -> n

3 kleiner 3 -> n

3 größer 1 -> j

0 kleiner 2 -> j

0 größer 1 -> n

1 kleiner 2 -> j

1 größer 2 -> n

2 kleiner 2 -> n

2 größer 1-> j

0 kleiner 1 -> j

0 größer 1 -> n

1 kleiner 1 -> n

1 größer 1 -> n