Fakultät IV Elektrotechnik und Informatik
Einführung in die Programmierung WS 2022/2023
Manfred Hauswirth
Damien Foucard / Uwe Kuehn / Aljoscha Meyer

Programmierblatt 09

Ausgabe: *19.01.2023 16:00*
Abgabe: *31.01.2023 08:00*

Thema: *Quicksort auf einfach verketteten Listen*

Abgabemodalitäten

1. Alle abzugebenden Quelltexte müssen ohne Warnungen und Fehler auf Deinem Rechner mit dem Befehl `clang -std=c11 -Wall -g` kompilieren.
2. Die Abgabe für den Quellcode erfolgt ausschließlich über unser Git im entsprechenden Branch. Nur wenn ein Ergebnis im ISIS-Kurs angezeigt wird, ist sichergestellt, dass die Abgabe erfolgt ist. Die Abgabe ist bestanden, wenn Du an Deinem Test einen grünen Haken siehst.
3. Du kannst bis zur Abgabefrist beliebig oft neue Versionen abgeben. Lies Dir die Hinweise der Tests genau durch, denn diese helfen Dir Deine Abgabe zu korrigieren.
Bitte beachte, dass ausschließlich die letzte Abgabe gewertet wird.
4. Die Abgabe erfolgt, sofern nicht anders angegeben, in folgendem Branch: `iprg-b<xx>-a<yy>`, wobei `<xx>` durch die zweistellige Nummer des Aufgabenblattes und `<yy>` durch die entsprechende Nummer der Aufgabe zu ersetzen sind.
5. Gib für jede Aufgabe die Quellcodedatei(en) gemäß der Vorgabe ab. Im ISIS-Kurs werden zum Teil Vorgabedateien bereitgestellt. Nutze diese zur Lösung der Aufgaben.
6. Die Abgabefristen werden vom Server überwacht. Versuche Deine Abgabe so früh wie möglich zu bearbeiten. Du minimierst so auch das Risiko, die Abgabefrist auf Grund von „technischen Schwierigkeiten“ zu versäumen. Eine Programmieraufgabe gilt als bestanden, wenn alle bewerteten Teilaufgaben bestanden sind.
7. Sofern die Aufgabenstellenstellung nichts gegenteiliges besagt, dürfen keine weiteren `include` Direktive verwendet werden, d.h., es dürfen keine zusätzlichen Libraryfunktionen verwendet werden. Eigene Funktionen zu implementieren und verwenden ist hingegen legitim und häufig eine gute Idee für besser lesbaren Code.

Aufgabe 1 Implementierung von Quicksort auf verketteten Listen (bewertet)

Implementiere die stabile Variante des in der Vorlesung vorgestellten Sortieralgorithmus `quick_sort` in C. Deine Funktion soll als Eingabe eine verkettete Liste bekommen und die Elemente sortiert zurückgeben.

Dazu findest du hier den Pseudocode für Quicksort, der seine generelle Funktionalität darstellt. Erarbeite dir die Funktionalität der Funktion `partition()` aus den Vorlesungsfolien. Das von `partition()` zurückgegebene Pivotelement muss für diese Aufgabe das erste Element der Eingabeliste sein.

Listing 1: Pseudocode Quicksort Algorithmus (rekursiv)

```
1  QuickSort (list tosort)
2      if (tosort.first == tosort.last)
3          // Listen mit 0 oder 1 Element(en) sind per
4          // Definition sortiert
5      else
6          list left, right
7          pivot ← Partition (tosort, left, right)
8
9          QuickSort(left)
10         QuickSort(right)
11
12         if (left.first == NULL)
13             tosort.first ← pivot
14         else
15             tosort.first ← left.first
16             left.last.next ← pivot
17
18         if (right.first == NULL)
19             pivot.next ← NIL
20             tosort.last ← pivot
21         else
22             pivot.next ← right.first
23             tosort.last ← right.last
```

Hinweis: Beachte, dass beim Einfügen in die rechte bzw. linke Teilliste kein neuer Speicher alloziert werden muss, sondern die jeweiligen `next`-Pointer entsprechend gesetzt werden.

Hinweis: Beachte für die Implementierung der Funktion `partition()`, dass sobald in der Liste weitere Elemente vorkommen, die denselben Wert wie das Pivotelement haben, diese in die rechte Teilliste eingefügt werden.
Das Pivotelement selbst ist nicht Teil der rechten bzw. linken Teilliste (s. Vorlesungsfolien), sondern tritt nur als Rückgabewert der Funktion `partition` auf.

Die Eingabedatei enthält eine Liste der beliebtesten Passwörter und deren Häufigkeit im Format:
`Passwort Häufigkeit`

Lies die Eingabedatei ein und speichere die Passwörter und ihre Häufigkeiten gemeinsam als Element einer einfach verketteten Liste ab. Dein Programm bekommt den Pfad zur Eingabedatei als erstes Argument.

Deine `quick_sort` Implementierung nimmt diese Liste als Eingabe und gibt am Ende die Passwort-Häufigkeits-Paare nach aufsteigender Häufigkeit sortiert aus. Das Ausgabeformat soll dabei dem Format der Eingabedatei entsprechen. Insbesondere werden also Paare durch Zeilenumbrüche getrennt, aber *vor dem ersten* Paar gibt es keinen Umbruch (nach dem letzten Paar aber schon). Beispiel: die Ausgabe für eine Liste aus zwei Paaren `foo 2` und `bar 2` müsste `"foo 2\nbar 3\n"` sein.

Nutze zur Lösung der Aufgabe die Vorgaben aus unserem ISIS-Kurs. Füge Deine Lösung als Datei `introprog_quicksort.c` im entsprechenden Abgabebranch in Dein persönliches Repository ein und übertrage die Lösung an die Abgabeplattform.