Fakultät IV - Elektrotechnik und Informatik
Einführung in die Programmierung WS 2015/16
Feldmann / Semmler / Lichtblau / Streibelt / Pujol / Rost

Aufgabenblatt 8

letzte Aktualisierung: 06. Januar, 15:12 Uhr

(eff804e590037f502980fd82115f6f0d9a205370)

Ausgabe: *Mittwoch, 6.1.2016*
Abgabe: *spätestens Freitag, 15.1.2016, 20:00*

Thema: *Korrektheit*

Abgabemodalitäten

- Alle abzugebenden Quelltexte müssen ohne Warnungen und Fehler auf den Rechnern des tubIT/IRB mittels `gcc -std=c99 -Wall` kompilieren.
- Abgaben erfolgen prinzipiell immer in Gruppen à 2 Personen, welche in den Tutorien festgelegt wurden. Einzelabgaben sind explizit als solche gekennzeichnet.
- Die Abgabe erfolgt ausschließlich über SVN. Die finale Abgabe
  - für Gruppenabgaben erfolgt im Unterordner `Tutorien/t<xx>/Gruppen/g<xx>/Abgaben/Blatt<xx>/`
  - für Einzelabgaben erfolgt im Unterordner `Tutorien/t<xx>/Studierende/<tuBIT-Login>/Abgaben/Blatt<xx>/`

  wobei die Ordner von uns erstellt werden.
- Benutze für alle Abgaben von Programmcode das folgende Namensschema: `introprog_blatt0X_aufgabe0Y_Z.c`, wobei X durch die Blattnummer, Y durch die Aufgabe und Z durch die Unteraufgabe zu ersetzen ist.
  Beispiel: Aufgabe 1.2 wird zu: `introprog_blatt01_aufgabe01_2.c`
  Gib für jede Unteraufgabe maximal eine Quellcodedatei ab, es sei denn, die Aufgabenstellung erfordert explizit die Abgabe mehrerer Dateien pro Aufgabe.

  Benenne alle anderen Abgaben (Pseudocode, Textaufgaben) wie oben beschrieben. Die zugelassenen Abgabeformate sind PDF, ODT und Text (txt). Verwende auch hier eine Datei pro Aufgabe, nicht jedoch pro Unteraufgabe.

1. Aufgabe: Korrektheit Bubblesort (3 Punkte)

In Listing 1 findet ihr den Pseudocode von Bubblesort, welcher ein übergebenes Array aufsteigend sortiert. Die Funktion `swap(A,i,j)` vertauscht hierbei die Werte in A, welche an der Stelle i und j stehen.

Listing 1: Bubblesort

```
1 BubbleSort(Array A)
2   for j ← length(A) downto 2 do
3       for i ← 1 to j−1 do
4           if A[i] > A[i+1] then swap(A,i,i+1)
```

Zeigt die Korrektheit unter Verwendung folgender Struktur:

A-1 Gebt die Schleifeninvariante für die äußere Schleife an. Hinweis: Aus dieser muss sich die Korrektheit des Algorithmus nach der Durchführung der äußeren Schleife ableiten lassen (siehe C-1).

A-2 Zeigt die initiale Gültigkeit der äußeren Schleifeninvariante, d.h. dass die Aussage nach der Initialisierung von j = length(A) in Zeile 2 gilt.

B-1 Gebt eine Schleifeninvariante für die innere Schleife an. Hinweis: Aus dieser Invariante muss sich letztlich die Korrektheit der äußeren Schleifeninvariante folgern lassen.

B-2 Zeigt die initiale Gültigkeit der inneren Schleifenvariante, d.h., dass die Invariante für jedes i = 1 gilt.

B-3 Zeigt, dass die Ausführung des inneren Schleifenkörpers (Zeile 4) die Gültigkeit der inneren Schleifeninvariante erhält.

A-3 Zeigt, dass die Gültigkeit der äußeren Schleifeninvariante durch die Ausführung der inneren Schleife (Zeile 3-4) erhalten bleibt. Benutzt dabei, dass die innere Schleifeninvariante beim Austritt aus der inneren for-Schleife gegolten hat.

C-1 Zeigt unter Verwendung der äußeren Schleifeninvariante die Korrektheit des Algorithmus, also dass nach dem Verlassen der äußeren Schleife alle Elemente in A aufsteigend sortiert sind.

Beachtet die folgenden Hinweise:

- Bitte gebt eure Abgabe als `introprog_blatt08_aufgabe01.pdf`, `introprog_blatt08_aufgabe01.odt` oder `introprog_blatt08_aufgabe01.txt` im entsprechenden Abgabeordner im SVN ab. Während wir nicht erwarten, dass ihr eure Abgabe mit Latex erstellt, beachtet bitte, dass ihr mathematisch präzise Aussagen treffen müsst.
- Die obige Struktur soll den roten Faden des Beweises verdeutlichen. Ihr fangt an Aussagen über die äußere Schleifeninvariante zu beweisen (A-1 und A-2), benötigt dann jedoch eine innere Schleifeninvariante und einen entsprechenden Beweis (B-1 bis B-3), um abschließend die Korrektheit der äußeren Schleifeninvariante (A-3) zu zeigen. Dies erlaubt euch die Korrektheit zu beweisen (C-1).
- Die Verwendung obiger Beweis-Struktur (A-1 bis A-3, B-1 bis B-3 und C-1) ist verpflichtend. Es muss klar ersichtlich sein, welcher Teil der Abgabe sich auf welches Beweis-Ziel bezieht.
- Bei der Angabge der Schleifeninvarianten A-1 und B-1 soll jeweils nur die jeweilige Invariante genannt werden. Diese Punkte bedürfen somit keiner Begründung. Die Schleifeninvarianten müssen jedoch mathematisch eindeutig definiert sein.
- Beachtet, dass die Invarianten A-1 und B-1 aus mehreren logischen Aussagen sowie Bedingungen in Abhängigkeit der jeweiligen Laufvariablen bestehen können.

- Du findest in ISIS den Korrektheitsbeweis für den verwandten SelectionSort Algorithmus als Orientierungshilfe. Dieser Beweis soll euch als Hilfe dienen und zeigen, wie Beweise mathematisch präzise durchgeführt wird. Wie in dem Beweis der Korrektheit von SelectionSort benötigt ihr bei den Unterpunkten A-2, A-3, B-2, B-3 und C-1 nicht viel Text schreiben, sondern müsst möglichst präzise und nachvollziehbar argumentieren.
- Es gibt keinerlei automatische Tests eurer Abgaben.
- Korrektheitsbeweise bestehen normalerweise aus zwei Teilen: (partieller) Korrektheit und der Terminierung. Die (partielle) Korrektheit besagt, dass das Ergebnis eines Algorithmus korrekt sein muss, wenn ein solches zurückgeliefert wird. Wird neben der partiellen Korrektheit auch bewiesen, dass der Algorithmus immer, d.h. unter jeglicher Eingabe, terminiert so spricht man von totaler Korrektheit, da in diesem Fall (1) immer ein Ergebnis zurückgeliefert wird und (2) dieses Ergebnis immer korrekt ist. Im Kontext der (iterativen) Sortieralgorithmen ist der Beweis der Terminierung einfach, da die Algorithmen nur aus for-Schleifen bestehen. Per Definition wird eine for-Schleife nur endlich oft durchgeführt, sofern die Lauf-Variable nicht andersweitig verändert wird. Der Schleifenkörper der for-Schleife `for i ← 1 to length(A) do ...` wird somit genau length(A) oft ausgeführt.