Fakultät IV - Elektrotechnik und Informatik
Einführung in die Programmierung WS 2014/15
Feldmann / Rost / Streibelt / Lichtblau / Deimel / Lucas

Aufgabenblatt 3

letzte Aktualisierung: 19. November, 12:18 Uhr

(d14804cd86908aab1e6d69634a66565f5974f092)

Ausgabe: Mittwoch, 19.11.2014
Abgabe: spätestens Freitag, 28.11.2014, 18:00

Thema: *Korrektheit von Selection Sort, Komplexitätsklassen, verkettete Listen*

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>/Blatt<xx>/submission/`
  - für Einzelabgaben erfolgt im Unterordner `Tutorien/t<xx>/Studierende/<tuBIT-Login>/Blatt<xx>/submission/`
- Benutzen Sie 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 ersetzen ist.
  Beispiel: Aufgabe 1.2 wird zu: `introprog_blatt01_aufgabe01_2.c`
  Für jede Unteraufgabe geben Sie maximal eine Quellcodedatei ab, es sei denn, die Aufgabenstellung erfordert explizit die Abgabe mehrerer Dateien pro Aufgabe.

  Alle anderen Abgaben (Pseudocode, Textaufgaben) benennen Sie wie oben beschrieben. Die zugelassenen Abgabeformate sind PDF, ODT und Text (txt). Auch hier verwenden Sie eine Datei pro Aufgabe, nicht jedoch pro Unteraufgabe.

1. Aufgabe: Korrektheit von Selectionsort (12 Punkte)

Hinweis: Geben Sie Ihre Lösung der Aufgabe in einem Textdokument mit einem der folgenden Namen ab: `introprog_blatt03_aufgabe01.{txt|odt|pdf}`

Gegeben ist der Pseudocode des Selection Sort Algorithmus:

```
1  SelectionSort(Array A)
2      for i←1 to length(A)−1 do
3          min←i
4          for j←i+1 to length(A) do
5              if A[j] < A[min] then
6                  min←j
7          tmp ← A[i]
8          A[i] ← A[min]
9          A[min] ← tmp
10     return A
```

1.1. (6 Punkte) Überlegen Sie sich wie sich der Inhalt des Arrays während der Ausführung des Algorithmus verändert. Formulieren Sie sich dann eine Schleifeninvariante, die angibt, welche Zahlen im Array schon auf jeden Fall an der richtigen Position sind und was über die Relation der restlichen Zahlen zu den bereits sortierten Zahlen gesagt werden kann.

1.2. (6 Punkte) Beweisen Sie die Korrektheit von Selectionsort. Zeigen Sie dabei zunächst, dass Ihre Schleifeninvariante bereits vor dem ersten Durchlauf der Schleife gilt. Zeigen Sie dann, dass die Schleifeninvariante auch nach jedem Durchlauf der Schleife weiterhin gilt, wenn sie zuvor gültig war und führen Sie einen induktiven Beweis.

2. Aufgabe: $\mathcal{O}$-Notation (13 Punkte)

Hinweis: Geben Sie Ihre Lösung der Aufgabe in einem Textdokument mit einem der folgenden Namen ab: `introprog_blatt03_aufgabe02.{txt|odt|pdf}`

2.1. (4 Punkte) Geben Sie, falls möglich, ein solches c an, sodass für alle $n \geq 1$ gilt, $f(n) \leq cg(n)$.

- a) $f(n) = 1000n$ und $g(n) = n^2$
- b) $f(n) = n^3 + n^2$ und $g(n) = n^3$
- c) $f(n) = 2^n$ und $g(n) = n^2$
- d) $f(n) = 10^{10} \log n$ und $g(n) = n$

2.2. (3 Punkte) Im folgenden sind 4 Funktionen gegeben. Geben Sie an, welche beide Funktionen a, b jeweils $a \in \Theta(b)$ und $b \in \Theta(a)$ erfüllen.

Geben Sie die Komplexitätsklassen mittels $\mathcal{O}$ möglichst genau an und ordnen Sie die Funktionen entsprechend aufsteigend.

- $f(n) = 10n^2$
- $g(n) = n^{0.1n}$
- $h(n) = n \log(10000n)$
- $i(n) = n + (0.00001n)^2 + 10$

2.3. (4 Punkte) Beweisen oder widerlegen Sie folgende Aussage: $f(n) = 100^{100}\sqrt{n} + n$ ist Element von $\mathcal{O}(n)$.

2.4. (2 Punkte) Betrachten Sie nun die Funktion $f(n) = 100^{100}\sqrt{n} + n + x^2$. Gilt Ihre Aussage aus der vorherigen Aufgabe auch für diese Funktion - liegt sie also in $\mathcal{O}(n)$? Begründen Sie Ihre Aussage; ein formaler Beweis ist nicht nötig.

3. Aufgabe: Implementierung der verketteten Liste (14 Punkte)

In dieser Aufgabe sollen Sie eine Funktion implementieren, die eine verkettete Liste und einen Schwellwert übergeben bekommt und zwei verkettete Listen erzeugt. Eine Liste soll dabei alle Zahlen enthalten, die kleiner als der Schwellwert sind. Die andere ausgegebene Liste soll alle Elemente enthalten, die größer oder gleich dem Schwellwert sind. Die Listenelemente der Ursprungsliste sollen dabei nicht kopiert werden, sondern für die beiden neuen Listen verwendet werden. Die Elemente der Liste sollen dabei die Reihenfolge aus der Ursprungsliste beibehalten.

3.1. (8 Punkte) Implementieren Sie die Funktion `threshold()` mit der oben beschriebenen Funktionalität für unsortierte, einfach verkettete Listen. Nutzen Sie dabei die Listenstruktur aus der Vorgabe.

Geben Sie den Quelltext dieser Unteraufgabe in einer Datei mit folgendem Namen ab:
`introprog_blatt03_aufgabe03_1.c`

Das kompilierte Programm können Sie wie folgt aufrufen:
`./program <Schwellwert> < seq_1.txt`

Wobei `<Schwellwert>` durch eine Zahl ersetzt werden muss.

Benutzen Sie folgende Codevorgabe:

Listing 1: Vorgabe `semester_blatt03_aufgabe03_1_vorgabe.c`

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <assert.h>
4
5  typedef struct list list;
6  typedef struct list_element list_element;
7
8  struct list
9  {
10   list_element *first;
11 };
12
13 struct list_element
14 {
15   int value;
16   list_element *next;
17 };
18
19
20 // Gibt die Liste nach dem gewünschten Format aus.
21 void print_liste (list * mylist)
22 {
23   list_element *curr_element = mylist->first;
24   while (curr_element != NULL)
25   {
26     printf (" %d", curr_element->value);
27     curr_element = curr_element->next;
28   }
29   printf ("\n");
30 }
31
32 int lese_liste (list * mylist)
33 {
```



```
34   int i = 0;
35   int temp;
36   list_element **ptr_to_last_next_ptr = &mylist->first;
37   while (scanf ("%d", &temp) == 1)
38   {
39     list_element *new_element = malloc (sizeof (list_element));
40     assert (new_element != NULL);
41     new_element->value = temp;
42     *ptr_to_last_next_ptr = new_element;
43     ptr_to_last_next_ptr = &(new_element->next);
44     i++;
45   }
46   *ptr_to_last_next_ptr = NULL;
47   return i;
48 }
49
50 void liste_freigeben (list * mylist)
51 {
52   list_element *curr = mylist->first;
53   while (curr != NULL)
54   {
55     list_element *next = curr->next;
56     free (curr);
57     curr = next;
58   }
59 }
60
61 void threshold (list * input, int threshold, list * output_smaller,
62   list * output_greaterequal)
63 {
64   // Implementieren Sie hier die threshold Funktion
65 }
66
67 int main (int argc, char *argv[])
68 {
69   int schwellwert;
70   if (argc != 2)
71   {
72     printf("Übergeben Sie den Schwellwert als Kommandozeilenparameter!\n");
73     return -1;
74   }
75   schwellwert = atoi (argv[1]);
76   list input;
77   list output_s;
78   list output_geq;
79   input.first = NULL;
80   output_s.first = NULL;
81   output_geq.first = NULL;
82   int len = lese_liste (&input);
83   printf ("Eingabeliste: ");
84   print_liste (&input);
85   threshold (&input, schwellwert, &output_s, &output_geq);
86   printf ("Unter dem Schwellwert: ");
87   print_liste (&output_s);
```

```
  printf ("Über dem Schwellwert: ");
  print_liste (&output_geq);
  printf ("Alte Eingangsliste: ");
  print_liste (&input);
  liste_freigeben (&input);
  liste_freigeben (&output_s);
  liste_freigeben (&output_geq);
  return 0;
}
```

3.2. (6 Punkte) Implementieren Sie die Funktion threshold_sorted() mit der oben beschriebenen Funktionalität für *sortierte*, einfach verkettete Listen, nutzen Sie dabei die vorgegebene Listenstruktur aus der Vorgabe. Nutzen Sie dabei aus, dass die Eingangsliste schon aufsteigend sortiert ist, um die Funktionalität möglichst effektiv zu realisieren.

Geben Sie den Quelltext dieser Unteraufgabe in einer Datei mit folgendem Namen ab:
`introprog_blatt03_aufgabe03_2.c`

Das kompilierte Programm können Sie wie folgt aufrufen:
`./program <Schwellwert> < seq_2.txt`

Wobei `<Schwellwert>` durch eine Zahl ersetzt werden muss.

Benutzen Sie folgende Codevorgabe (Anfang identisch mit vorheriger Aufgabe):

Listing 2: Vorgabe `semester_blatt03_aufgabe03_2_vorgabe.c`

```
  // Implementieren Sie hier die threshold Funktion
}

int main (int argc, char *argv[])
{
  int schwellwert;
  if (argc != 2)
  {
    printf("Übergeben Sie den Schwellwert als Kommandozeilenparameter!\n");
    return -1;
  }
  schwellwert = atoi (argv[1]);
  list input;
  list output_s;
  list output_geq;
  input.first = NULL;
  output_s.first = NULL;
  output_geq.first = NULL;
  int len = lese_liste (&input);
  printf ("Eingabeliste: ");
  print_liste (&input);
  threshold_sorted (&input, schwellwert, &output_s, &output_geq);
  printf ("Unter dem Schwellwert: ");
  print_liste (&output_s);
  printf ("Über dem Schwellwert: ");
  print_liste (&output_geq);
  printf ("Alte Eingangsliste: ");
  print_liste (&input);
  liste_freigeben (&input);
```



```
  liste_freigeben (&output_s);
  liste_freigeben (&output_geq);
  return 0;
}
```