Fakultät IV - Elektrotechnik und Informatik
IntroProg WS 2014/15, Feldmann / Rost / Streibelt / Deimel / Lucas

Aufgabenblatt 2

letzte Aktualisierung: 17. November, 12:33 Uhr

(1aa1c2a6c0f08a4253f9f4f45b72595de5a031be)

Ausgabe: *Mittwoch, 12.11.2014*
Abgabe: *spätestens Freitag, 21.11.2014, 18:00*

Thema: *Laufzeitanalyse, Selection Sort*

Abgabemodalitäten

- Alle abzugebenden Quelltexte müssen ohne Warnungen und Fehler auf den Rechnern des tubIT/IRB mittels `gcc -std=c99 -Wall` kompilieren.
- Die Abgabe erfolgt ausschließlich über SVN. Die finale Abgabe
  - Für Einzelabgaben erfolgt im Unterordner `Tutorien/t<xx>/Studierende/<tuBIT-Login>/Blatt<xx>/submission/`
  - Für Gruppenabgaben erfolgt im Unterordner `Tutorien/t<xx>/Gruppen/g<xx>/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`
  Bei Abgaben in Textform (Pseudocode, Textaufgaben) benennen Sie die Dateien ebenfalls nach diesem Namensschema. Zulässige Abgabeformate sind PDF, ODT und Plaintext (txt).

  Für jede Unteraufgabe geben Sie maximal eine Quellcodedatei und maximal ein Textdokument ab, es sei denn, die Aufgabenstellung erfordert explizit die Abgabe mehrerer Dateien pro Aufgabe.

1. Aufgabe: Laufzeitanalyse von `finde_minimum()` (13 Punkte)

Der folgende Pseudocode findet das kleinste Element einer Liste:

```
1  FindeMinimum(Array A)
2      min←A[1]
3      for i←2 to length(A) do
4          if A[i] < min then
5              min←A[i]
6      return min
```

1.1. (10 Punkte) Führen Sie eine Laufzeitanalyse des Pseudocodes durch. Halten Sie sich dabei an die in der Vorlesung vorgestellte Methode und Notation. Geben sie insbesondere folgende Ergebnisse an:

- Anzahl der benötigten Zeitschritte jeder Pseudocode Zeile
- Gesamtzahl der benötigten Zeitschritte des Algorithmus

1.2. (3 Punkte) Bestimmen Sie die Worst Case Laufzeitkomplexität des Algorithmus. Verwenden sie die $\mathcal{O}$-Notation.

Geben Sie die Ihre vollständige Analyse und deren Ergebnisse in einem Textdokument ab:
`introprog_blatt02_aufgabe01.{txt|odt|pdf}`

2. Aufgabe: Laufzeitanalyse des Selection Sort Algorithmus (13 Punkte)

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
```

2.1. (10 Punkte) Führen Sie eine Laufzeitanalyse des Pseudocodes durch. Halten Sie sich dabei an die in der Vorlesung vorgestellte Methode und Notation. Geben sie insbesondere folgende Ergebnisse an:

- Anzahl der benötigten Zeitschritte jeder Pseudocode Zeile
- Gesamtzahl der benötigten Zeitschritte des Algorithmus

2.2. (3 Punkte) Bestimmen Sie die worst case Laufzeitkomplexität des Algorithmus. (Wie viele Zeitschritte werden im ungünstigsten Fall ausgeführt?) Verwenden sie die $\mathcal{O}$-Notation.

Geben Sie die Ihre vollständige Analyse und deren Ergebnisse in einem Textdokument ab:
`introprog_blatt02_aufgabe02.{txt|odt|pdf}`

3. Aufgabe: Implementierung des Selection Sort Algorithmus (16 Punkte)

3.1. (10 Punkte) Implementieren Sie den Pseudocode in einer C Funktion namens `selection_sort()`. Halten Sie sich dabei an die Pseudocode Vorlage! Als Argumente nimmt die Funktion die Startadresse und Länge eines Integer Arrays. Nach dem Aufruf sind die Elemente der Array aufsteigend sortiert.

3.2. (6 Punkte) Verwenden Sie die Sortierfunktion in einem lauffähigen Programm, das eine Liste von Integer Zahlen über die Standardeingabe einliest, und diese sortiert über die Standardausgabe ausgibt.

Tipp: Die Liste von Integer Zahlen kann über eine sogenannte Dateiumleitung (Redirection) aus einer Datei mitgegeben werden. Das ist genauso als würden die Zahlen über die Standardeingabe einzeln eingegeben und die Eingabe mit Ctrl-D beendet. Das Programm wird dazu wie folgt aufgerufen:
`./introprog_blatt02_aufgabe03 < zahlen.txt`

Geben Sie Ihren Quelltext als `introprog_blatt02_aufgabe03.c` ab. Halten sie sich dabei an folgende Codevorgabe:

Listing 1: Vorgabe `introprog_blatt02_aufgabe03_vorgabe.c`

```c
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

int MAX_LAENGE = 1000;

// Gibt das Array im korrekten Format aus.
void print_array(int array[], int len) {
    printf("Liste:");
    for (int i = 0; i < len ; i++) {
        printf(" %d", array[i]);
    }
    printf("\n");
}

//Liest eine Liste an Zahlen von der Standardeingabe und speichert diese
   ↪ in array[]
//Gibt die Anzahl der eingelesenen Zahlen zurück
int lese_array(int array[]) {
    int i = 0;
    while (scanf("%d", &array[i]) == 1) {
        i++;
        assert(i<MAX_LAENGE); //Bricht das Programm lautstark ab, wenn
           ↪ diese Annahme verletzt wird.
    }
    return i;
}


/***************** Zu implementierende Funktionen ********************/

// Nach dem Aufruf ist das Array sortiert
void selection_sort(int array[], int len) {
    // HIER
    // Implementiere Selection Sort nach der Pseudocode Vorgabe
    // Markiere in Kommentaren die Pseudocodezeilennummern, die in der
       ↪ jeweiligen Quelltextzeile implementiert wird.
}


int main(int argc, char *argv[]) {
    int array[MAX_LAENGE];
    int len = lese_array(array);
    printf("Eingabe:\n");
    print_array(array, len);
    //HIER rufe selection_sort auf
    printf("Sortiert:\n");
    print_array(array, len);
    return 0;
}
```