Fakultät IV - Elektrotechnik und Informatik
Einführung in die Programmierung WS 2017/18
Feldmann / Foucard / Lichtblau / Streibelt / Krüger / Enghardt / Palmer

Aufgabenblatt 1

letzte Aktualisierung: 10. November, 11:45 Uhr

(28daf9b595da1a85e03c2a697738fa01cc62e2fc)

Ausgabe: Freitag, 10.11.2017
Abgabe: spätestens Mittwoch, 22.11.2017, 18:00

Thema: *Insertion Sort, Count Sort, Pseudocode*

Abgabemodalitäten

- Alle abzugebenden Quelltexte müssen ohne Warnungen und Fehler auf den Rechnern des eecsIT mittels `gcc -std=c99 -Wall` kompilieren.
- Für die Hausaufgaben im Semester gibt es ein eigenes Subversion-Repository (SVN). Bitte die Hinweise auf Blatt 1 dazu beachten!
- Die Abgaben für Blatt 1 bis 4 erfolgen als Einzelabgaben. Ab Blatt 5 erfolgen Abgaben in festen Gruppen à 2 Personen.
- Die Gruppen werden vor Ausgabe von Blatt 5 gebildet. Solltet ihr keine Gruppe gebildet haben, werdet ihr einer Gruppe zugewiesen.
- Alle Abgaben müssen an der dafür vorgesehenen Stelle die Namen aller Gruppenmitglieder bzw. bei Einzelabgaben des Autors/der Autorin enthalten!
- Die Abgabe erfolgt ausschließlich über unser SVN im Abgaben-Ordner. 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>/`

  WICHTIG: Die Abgabeordner werden immer von uns erstellt, sonst kommt es zu Konflikten! Benutzt zum Aktualisieren `svn up` im obersten Verzeichnis des Repositories!
- Benutzt für alle Abgaben die in der jeweiligen Aufgabe angegebenen Dateinamen (die von uns vorgegebenen Dateien haben das Wort "vorgabe" im Dateinamen, du musst die Datei mit deiner Lösung also entsprechend umbenennen.)
- Gib bei den Programmieraufgaben den C-Quellcode ab und achte darauf, dass die Datei mit .c endet
- Bei Textaufgaben sind, wenn nicht anders angegeben, .txt Dateien zugelassen, die als Plaintext-Datei zu speichern sind. (Keine Word-Dateien umbenennen, etc.!)

Registrierung für das Modul und die Hausaufgaben

Um Hausaufgaben abgeben zu können, musst Du Dich für die Modulprüfung angemeldet haben, da es sich um eine Portfolioprüfung handelt. Erst dann kannst Du Abgaben im SVN hochladen.

Die Anmeldung erfolgt je nach Studiengang unterschiedlich, leider müssen wir die Anmeldungen auch unterschiedlich erfassen und euch daher bitten die folgenden Vorgaben einzuhalten.

Wenn Dir auf Osiris bereits eine Anmeldung zu den Hausaufgaben im Semester angezeigt wird, haben wir Deine Anmeldung erfolgreich entgegengenommen, siehe: https://teaching.inet.tu-berlin.de/services/osiris-wise1718/signup/courses/course-1/do-view

Die meisten sollten die **Anmeldung in QISPOS für "Einf.i.d.Programmierung", Prüfungsnummer 6135** nutzen können. Wenn Du QISPOS nutzen kannst melde Dich für den Termin am 21.02.2018 an, auch wenn Du den zweiten Test statt des Ersten schreiben möchtest. DIE FRIST DAZU ENDET AM 13. NOVEMBER. Nach erfolgreicher Anmeldung in QISPOS musst Du nichts weiter machen, wir übernehmen die Daten aus QISPOS.

Wenn Du **im 3. Versuch (mündliche Prüfung)** bist, sende eine Email mit dem Betreff '3. Versuch' an rt+introprog-team@inet.tu-berlin.de. Die Email sollte Deine Matrikelnummer und dein tubit-Login enthalten. Du kannst dann das Testsystem in Osiris benutzen, um Aufgaben abzugeben, diese dürfen wir aber nicht bewerten. Gruppenabgaben kannst Du nur als 'Einergruppe' abgeben. Einen Termin für die mündliche Prüfung musst Du in jedem Fall gesondert in einer E-Mail an rt+re-exam@inet.tu-berlin.de vereinbaren und Dich im Prüfungsamt per Papierformular anmelden, dass Du in unserem Sekretariat rechtzeitig vor der Prüfung abgibst.

Studierende von **MintGrün, Schülerstudierende, Seniorenstudenten** senden uns eine Email mit dem Betreff 'Modulanmeldung' an rt+introprog-team@inet.tu-berlin.de. In der Email sollte der tubit-Nutzername und ein Hinweis auf den Studiengang/das Studienprogramm und falls vorhanden die Matrikelnummer stehen.

Gasthörer des **In(2)TU** Programms müssen ihre Gasthörerkarte von Prof. Feldmann unterschreiben lassen. Dazu diese Karte in unserem Sekretariat abgeben. Wenn ihr bereits eine Unterschrift bekommen habt aber in Osiris noch keine Anmeldung zu den Hausaufgaben zu sehen ist, schickt bitte ein Foto von der unterschriebenen Karte zusammen mit eurem Namen an rt+introprog-team@inet.tu-berlin.de.

Alle anderen **Gasthörer** und **Nebenhörer** geben den 'Antrag auf Gasthörerschaft' bzw. 'Nebenhörerschaft' im Sekretariat MAR 4-4, Raum MAR 4.006 ab und senden eine Email mit dem Betreff 'Gasthörer' bzw. 'Nebenhörer' an rt+introprog-team@inet.tu-berlin.de.

Wenn Ihr Introprog als **Zusatzmodul** oder **Freie Wahl** einbringt, füllt das gelbe Formular 'PRÜFUNGSMELDUNG' im Prüfungsamt aus, lasst es dort bestätigen und bringt es in unser Sekretariat, Raum MAR 4.006 und sendet eine Email mit dem Betreff 'Zusatzmodul' bzw. 'Freie Wahl' an rt+introprog-team@inet.tu-berlin.de.

Alle anderen Studierenden füllen entweder das **gelbe Formular** 'Anmeldung zur Prüfung' oder 'PRÜFUNGSMELDUNG zur [] Bacherlorprüfung des Moduls....' im Prüfungsamt aus und geben dieses unterschrieben in unserem Sekretariat MAR 4-4, Raum MAR 4.006 ab und senden eine Email mit dem Betreff 'Modulanmeldung gelbes Formular' an rt+introprog-team@inet.tu-berlin.de.

Hinweise zum SVN-Repository für das Semester

Wie auch im C-Kurs wirst Du die Abgaben für die Vorlesung "Einführung in die Programmierung" im SVN abgeben. Dazu musst Du ein neues SVN-Repository auschecken. Dieses wird erst nach erfolgreicher Anmeldung für Dich freigeschaltet!
Wechsel mit dem Kommando `cd` in das Verzeichnis, wo das neue Repository liegen soll (NICHT im C-Kurs-Repository, sondern z.B. im Verzeichnis darüber), und gebe dann folgenden Befehl (in einer Zeile) ein:
`svn co --username <TUBIT-LOGIN>`
`https://teaching.inet.tu-berlin.de/services/svn/introprog-wise1718`
Nach einem erfolgreichen Checkout findest Du in diesem Repository alle Materialien aus dem C-Kurs im Ordner `C-Kurs`, sowie eine Verzeichnisstruktur, welche die Materialien und Aufgabenstellungen für den Rest des Semesters enthalten wird.
Die Verzeichnisse in diesem Repository sind nach Tutorien und innerhalb der Tutorien nach Arbeitsgruppen und Teilnehmern organisiert. Die Ordner werden wieder automatisch von uns angelegt, sobald Ihr angemeldet seid. Wir stellen auch wieder private Arbeitsverzeichnisse zur Verfügung - diese gibt es pro Gruppe und pro TeilnehmerIn, so dass Ihr auch in den Gruppen Arbeitsversionen der Aufgaben teilen und gemeinsam bearbeiten könnt.

Wichtig: Die Gruppenordner legen wir im SVN an, sobald die Gruppen gebildet wurden und Blatt 5 veröffentlicht wurde. Die **Gruppenbildung** wird voraussichtlich mit Blatt 4 beginnen und auf einem Arbeitsblatt erklärt werden. Derzeit könnt Ihr noch keine Gruppen bilden.

Denk daran, dass Du alle Änderungen die wir am Repository vornehmen (Abgabeordner, neue Materialien, Gruppenordner, ...) mittels `svn up` vom SVN-Server 'abholen' musst! Lege die Abgabeordner nicht selbst an, sonst wird es dabei zu Konflikten kommen!
Hinweis: Das Kommando `svn up` aktualisiert die Inhalte im aktuellen Verzeichnis inklusive aller Unterverzeichnisse, übergeordnete Verzeichnisse werden NICHT aktualisiert. Daher empfehlen wir, `svn up` immer im obersten Verzeichnis des Repositories auszuführen.

1. Aufgabe: Implementierung Insertion Sort (1 Punkte)

Implementiere anhand des Pseudocodes in Listing 1 und den Vorlesungsfolien die Funktion `insertion_sort()` in C. Beachte dabei, dass per Konvention Arrayindizes in Pseudocode bei 1 beginnen, in C jedoch bei 0! Passe die Indizes in deiner Implementierung also entsprechend an!

Listing 1: Pseudocode Insertion Sort

```
1 InsertionSort(Array A)
2     for j←2 to length(A) do
3         key←A[j]
4         i←j−1
5         while i>0 and A[i]>key do
6             A[i+1] ← A[i]
7             i ← i−1
8         A[i+1] ← key
```

Die Funktion bekommt als Eingabeparameter ein Integer-Array, das sortiert zurückgegeben werden muss. Die zu sortierenden Zahlen werden aus einer Datei eingelesen. Verwende dazu die, in der Datei `arrayio.c` vorgegebene, Funktion `read_array_from_file`. Gib am Ende deines Programms das sortierte Array mit der Funktion `print_array` aus.

Hinweis: Mach dich mit der Signatur der vorgegebenen Funktionen vertraut, um diese korrekt aufzurufen:

- `int read_array_from_file(int array[], size_t size, char *filename)`
  // Diese Funktion liest eine Folge von maximal `size` vielen Zahlen aus der Datei mit dem Namen `filename` ein und fügt sie in das Array `array` ein.
- `void print_array(int array[], int len)`
  // Diese Funktion gibt die ersten `len` vielen Einträge des Arrays `array` mittels printf aus.

Im SVN findest du eine Beispielliste mit zu sortierenden Zahlen in der Datei `zahlen_insertionsort.txt`. Die Werte in dieser Datei dienen nur als Beispiele. Teste dein Programm also mit unterschiedlichen Eingaben. Halte Dich dabei an das Format der Datei `zahlen_insertionsort.txt`.
Das folgende Listing zeigt dir einen beispielhaften Programmaufruf:

Listing 2: Programmbeispiel

```
1 > gcc -std=c99 -Wall introprog_insertionsort.c\\
2 arrayio.c -o introprog_insertionsort
3 > ./introprog_insertionsort zahlen_insertionsort.txt
4 Unsortiertes Array: 72 -100 1 10 23 -1 55 9 48 2 -53 5
5 Sortiertes Array: -100 -53 -1 1 2 4 9 10 23 48 55 72
```

Benutze die folgende Codevorgabe:

Listing 3: Vorgabe `introprog_insertionsort_vorgabe.c`

```
1  /* === INTROPROG ABGABE ===
2   * Blatt 1, Aufgabe 1
3   * Tutorium: tXX
4   * Abgabe von: Erika Mustermann
5   * ========================
6   */
7
8  #include <stdio.h>
9  #include <stdlib.h>
10 #include "arrayio.h"
11
12 int MAX_LAENGE = 1000;
13
14 void insertion_sort(int array[], int len) {
15     // HIER insertion sort implementieren
16     // Diese Funktion soll das Eingabearray nach dem Insertion Sort-Verfahren
17     // sortieren.
18     // Hinweis: Verwende die "in place"-Variante! D.h. der Sortiervorgang
19     // soll auf dem originalen Array stattfinden und kein zweites verwendet
20     // werden.
21 }
22
23 int main(int argc, char *argv[]) {
24
25     if (argc < 2){
26         printf("Aufruf: %s <Dateiname>\n", argv[0]);
27         printf("Beispiel: %s zahlen.txt\n", argv[0]);
28         exit(1);
```

```c
    }

    char *filename = argv[1];

    int array[MAX_LAENGE];
    int len = read_array_from_file(array, MAX_LAENGE, filename);

    printf("Unsortiertes Array:");
    print_array(array, len);

    // Aufruf insertion sort

    printf("Sortiertes Array:");
    print_array(array, len);

    return 0;
}
```

2. Aufgabe: Implementierung Count Sort (2 Punkte)

Implementiere anhand des Pseudocodes in Listing 4 und der Vorlesungsfolien die Funktion `count_sort()` in C. Beachte dabei, dass per Konvention Array-Indizes in Pseudocode bei **1** beginnen, in C jedoch bei **0**! Passe die Indizes in deiner Implementierung also entsprechend an!

Listing 4: Pseudocode Count Sort

```
CountSort(Array A_in, Array A_out)
    // C ist Hilfsarray mit 0 initialisiert
    for j←1 to length(A_in) do
        C[A_in[j]]←C[A_in[j]]+1

    k←1
    for j←1 to length(C) do
        for i←1 to C[j] do
            A_out[k]←j
            k←k+1
```

Die Funktion bekommt als Eingabeparameter zwei Integer-Arrays. Im ersten werden die zu sortierenden Werte gespeichert und im zweiten die nach Durchlauf des Algorithmus sortierte Folge von Werten.

Hinweis: Wir gehen zur Vereinfachung hier davon aus, dass nur Werte im Bereich $\{0, 1, \ldots, \texttt{MAX\_VALUE}\}$ sortiert werden sollen. Der Wert `MAX_VALUE` wird hierbei in der Vorgabe definiert[1].
Die zu sortierenden Zahlen werden aus einer Datei eingelesen. Verwende dazu die, in der Datei `arrayio.c` vorgegebene, Funktion `read_array_from_file`. Gib am Ende deines Programms das sortierte Array mit der Funktion `print_array` aus.

Hinweis: Mach dich mit der Signatur der vorgegebenen Funktionen vertraut, um diese korrekt aufzurufen (siehe die Erklärung von Aufgabe 1).
Im SVN findest du eine Beispielliste mit zu sortierenden Zahlen in der Datei `zahlen_count_sort.txt`. Die Werte in dieser Datei dienen nur als Beispiele. Teste dein Programm also mit unterschiedlichen Eingaben. Halte dich dabei an das Format der Datei `zahlen_insertion_sort.txt`.

[1] Du solltest deinen Code auch mit verschiedenen `MAX_VALUE` Werten testen; benutze im Code immer die Konstante `MAX_VALUE` anstatt einer festen Zahl wie 100.

Das folgende Listing zeigt dir einen beispielhaften Programmaufruf:

Listing 5: Programmbeispiel

```
> gcc -std=c99 -Wall introprog_countsort.c\\
arrayio.c -o introprog_countsort
> ./introprog_countsort zahlen_countsort.txt
Unsortiertes Array: 90 38 42 34 8 0 77 1 84 5 25 72 44 42 90 63 23
Sortiertes Array: 0 1 5 8 23 25 34 38 42 42 44 63 72 77 84 90 90
```

Benutze die folgende Codevorgabe:

Listing 6: Vorgabe `introprog_countsort_vorgabe.c`

```c
/* === INTROPROG ABGABE ===
 * Blatt 1, Aufgabe 2
 * Tutorium: tXX
 * Abgabe von: Erika Mustermann
 * ========================
 */

#include <stdio.h>
#include <stdlib.h>
#include "arrayio.h"

int MAX_LAENGE = 1000;
int MAX_VALUE = 100;

void count_sort_calculate_counts(int input_array[], int len, int count_array
    ↪ []) {
    // Hier Funktion implementieren
}

void count_sort_write_output_array(int output_array[], int len, int
    ↪ count_array[]) {
    // Hier Funktion implementieren
}



int main(int argc, char *argv[]) {

    if (argc < 2){
        printf("Aufruf: %s <Dateiname>\n", argv[0]);
        printf("Beispiel: %s zahlen.txt\n", argv[0]);
        exit(1);
    }

    char *filename = argv[1];

    int input_array[MAX_LAENGE];
    int len = read_array_from_file(input_array, MAX_LAENGE, filename);

    printf("Unsortiertes Array:");
    print_array(input_array, len);

    // HIER alle nötigen Deklarationen und Funktionsaufrufe für Count Sort
        ↪ einfügen

    printf("Sortiertes Array:");
    // Folgende Zeile einkommentieren, um das Array auszugeben
    // print_array(output_array, len);

    return 0;
}
```