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

Aufgabenblatt 1

letzte Aktualisierung: 12. November, 17:58 Uhr

(892160aa264e4c8bda2141ecbe10cb71a6634a76)

Ausgabe: *Mittwoch, 4.11.2015*
Abgabe: *spätestens Freitag, 13.11.2015, 16:00*

Thema: *Insertion Sort, Count Sort, Pseudocode*

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.

Wie auch im C-Kurs wirst Du die Abgaben für die Vorlesung ”Einführung in die Programmierung” im SVN abgeben. Dazu musst Du einmal ein neues SVN auschecken. Wechsel mit dem Kommando `cd` in das Verzeichnis, wo das neue Repository liegen soll und gebe dann folgenden Befehl ein:

`svn co https://teaching.inet.tu-berlin.de/tubitauth/svn/introprog-ws1516`

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 dabei nach Tutorien und innerhalb der Tutorien nach Arbeitsgruppen organisiert. Die Ordner werden wieder automatisch angelegt. Eine Ausnahme bilden wieder die Arbeitsverzeichnisse - diese gibt es pro Gruppe und pro TeilnehmerIn, so dass Du auch in den Gruppen Arbeitsversionen Deiner Aufgaben mit Deinem Gruppenpartnern gemeinsam bearbeiten kannst.

Wichtig: Die Gruppenordner legen wir im SVN an, sobald die Gruppen gebildet wurden, **nicht aber vor Freitag, dem 6. November, 15:00 Uhr**. Um diese und andere Änderungen unsererseits zu sehen, welchsel in das oberste Verzeichnis des Repositoies und führe das Kommando `svn up` aus. *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.

Hinweise zur Gruppenbildung

Aufgaben die als ’Gruppenabgabe’ gekennzeichnet sind werden grundsätzlich in festen Gruppen die aus zwei Studierenden bestehen gemeinsam bearbeitet und eingereicht.

Dazu musst Du selbstständig Gruppen bilden. Solltest Du keinen Gruppenpartner/Gruppenpartnerin finden, werden die Tutoren die Gruppen zuweisen.

Die Gruppenbildung selbst wird über das bekannte Webinterface in Osiris organisiert.

Besuche dazu folgende URL:
`https://teaching.inet.tu-berlin.de/tubitauth/osiris/topple/groups`

Hier kannst Du über den angezeigten grünen Button bis zu 2 Gruppenvorschläge anlegen und jeweils eine andere Person einladen mit Dir eine Gruppe zu bilden. Dazu benötigst Du den Benutzernamen der anderen Person und gebe diesen in der Maske ein. Die eingeladene Person bekommt dann in Osiris deinen Namen angezeigt sowie die Nachricht die Du angibst.

Wenn Du eingeladen wirst eine Gruppe zu bilden, siehst Du auf der oben genannten Seite eine Liste der eingegangenen Einladungen und kannst diese annehmen oder ablehnen sowie eine Nachricht angeben.

Sobald Du eine Einladung angenommen hast oder eine Person die Du eingeladen hast Deine Einladung annimmt, wird automatisch die Gruppe erzeugt. Der Gruppenordner im SVN-Repository wird jedoch nicht vor Freitag Nachmittag erscheinen.

Es werden generell keine Emailbenachrichtigungen versandt, Du musst also selbst auf der Seite nachsehen, ob Du eingeladen wurdest.

Weitere Details findest Du auf Osiris während der Benutzung der Funktion.

Beachte bitte, dass nur Gruppen innerhalb eines Tutoriums gebildet werden können. Falls Du bereits einen Gruppenpartner hast der nicht in Deinem Tutorium ist, liegt es im Ermessen der Tutoren, ob Du das Tutorium tauschen kannst. Spreche die Tutoren dazu bitte im ersten Tutorium an!

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 `input_blatt01.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_insertion_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`.
Das folgende Listing zeigt dir einen beispielhaften Programmaufruf:

Listing 2: Programmbeispiel

```
1 > gcc -std=c99 -Wall introprog_blatt01_aufgabe01.c\\
2 input_blatt01.c -o introprog_blatt01_aufgabe01
3 > ./introprog_blatt01_aufgabe01 zahlen_insertion_sort.txt
4 Unsortiertes Array: 1 23 55 12 0 4 3 -1 5 -100
5 Sortiertes Array: -100 -1 0 1 3 4 5 12 23 55
```

Benutze die folgende Codevorgabe:

Listing 3: Vorgabe `introprog_blatt01_aufgabe01_vorgabe.c`

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "input_blatt01.h"
4
5  int MAX_LAENGE = 1000;
6
7  void insertion_sort(int array[], int len) {
8      // HIER insertion sort implementieren
9      // Diese Funktion soll das Eingabearray nach dem Insertion Sort Verfahren
10     // sortieren.
11     // Hinweis: Verwende die inplace Variante! D.h. der Sortiervorgang soll
           ↪ auf
12     // dem originalen Array stattfinden und kein zweites verwendet werden
13 }
14
15 int main(int argc, char *argv[]) {
16
17     if (argc < 2){
18         printf("Aufruf: %s <Dateiname>\n", argv[0]);
19         printf("Beispiel: %s zahlen.txt\n", argv[0]);
20         exit(1);
21     }
22
23     char *filename = argv[1];
24
25     int array[MAX_LAENGE];
26     int len = read_array_from_file(array, MAX_LAENGE, filename);
27
28     printf("Unsortiertes Array:");
29     print_array(array, len);
30
31     // Aufruf insertion sort
32
33     printf("Sortiertes Array:");
34     print_array(array, len);
35
36     return 0;
37 }
```


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

```
1  CountSort(Array A_in, Array A_out)
2      // C ist Hilfsarray mit 0 initialisiert
3      for j←1 to length(A_in) do
4          C[A_in[j]]←C[A_in[j]]+1
5
6      k←1
7      for j←1 to length(C) do
8          for i←1 to C[j] do
9              A_out[k]←j
10             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 `input_blatt01.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`.
Das folgende Listing zeigt dir einen beispielhaften Programmaufruf:

Listing 5: Programmbeispiel

```
> gcc -std=c99 -Wall introprog_blatt01_aufgabe02.c\\
input_blatt01.c -o introprog_blatt01_aufgabe02
> ./introprog_blatt01_aufgabe02 zahlen_count_sort.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_blatt01_aufgabe02_vorgabe.c`

```c
#include <stdio.h>
#include <stdlib.h>
#include "input_blatt01.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:");
    print_array(output_array, len);

    return 0;
}
```

[1] Du solltest deinen Code auch mit verschiedenen `MAX_VALUE` Werten testen; benutze immer den Wert `MAX_VALUE`, aber nie den derzeitigen Wert 100.