Fakultät IV Elektrotechnik und Informatik
Einführung in die Programmierung WS 2019/2020
Manfred Hauswirth / Damien Foucard / Christoph Seifert

Aufgabenblatt 1

Ausgabe: *08.11.2019*
Abgabe: *19.11.2019 09:59*

Thema: *Insertionsort, Countsort, Pseudocode*

Abgabemodalitäten

1. Alle abzugebenden Quelltexte müssen ohne Warnungen und Fehler auf den Rechnern des ZE-CM/eecsIT mittels `clang -std=c11 -Wall -g` kompilieren.
2. Die Abgabe erfolgt ausschließlich über unser SVN im Abgaben-Ordner. Nur wenn ein Test in Osiris angezeigt wird ist sichergestellt, dass die Abgabe erfolgt ist.
3. Du kannst bis zur Abgabefrist beliebig oft neue Versionen abgeben.
4. Die Abgabe erfolgt in folgendem Unterordner:
   `introprog-wise1920/Studierende/<L>/<TU-Login>/Abgaben/Blatt<XX>`
   wobei `<L>` durch den ersten Buchstabe des TU-Logins und `<XX>` durch die Nummer des Aufgabenblattes zu ersetzen sind. Die Ordner werden automatisch angelegt sobald die Abgabe freigeschaltet wird.
5. Du darfst den Abgabeordner für das Blatt nicht selbst erstellen. Die Ordner erstellen wir kurz nach der Ausgabe der Aufgabe.
6. Um die Änderungen (z. B. neue Abgabeordner) vom Server abzuholen, musst Du den Befehl `svn update` im Wurzelverzeichnis (also im Verzeichnis `introprog-wise1920`) des Repositories ausführen.
7. Im Abgabeordner gelten einige restriktive Regeln. Dort ist nur das Einchecken von Dateien mit den in der Aufgabe vorgegebenen Namen erlaubt, ausserdem werden die Abgabefristen vom Server überwacht. Beachte eventuelle Fehlermeldungen beim SVN-Commit. Lade nur Dateien hoch, die Du selbst bearbeiten sollst, insbesondere also keine Vorgaben.
8. Der Dateiname bei Abgaben mit C-Code soll dem der Vorgaben entsprechen. Entferne ausschließlich `_vorgabe` aus dem Name. Bei Textaufgaben sind, wenn nicht anders angegeben, `.txt` Dateien zugelassen, die als Plaintext-Datei zu speichern sind (keine Word-Dateien umbenennen).
9. Es gibt einen Ordner `Arbeitsverzeichnis`, in dem Du Dateien für Dich ablegen kannst.
10. Die Ergebnisse der automatischen Tests kannst Du auf OSIRIS einsehen:
    https://osiris.ods.tu-berlin.de/

Hinweise zum 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 das neue SVN-Repository auschecken. Dies ist allerdings erst nach der Anmeldung für die Prüfungskomponente “Hausaufgaben im Semester” in Osiris möglich.

Um das Repository auszuchecken, wechsel mit dem Kommando `cd` in ein beliebiges Verzeichnis, welches selbst in keinen Repository liegt (z. B. nicht im C-Kurs-Repository, sondern im Verzeichnis darüber), und führe folgenden Befehl aus (in einer Zeile):
`svn co --username <TUBIT-LOGIN>`
`https://osiris.ods.tu-berlin.de/svn/introprog-wise1920`

Denke daran, dass Du alle Änderungen die wir am Repository vornehmen (neue Abgabeordner oder Materialien) vom SVN-Server mittels `svn update` im Wurzelverzeichnis des Repositories abholen musst! Lege die Abgabeordner nicht selbst an, sonst entstehen Konflikte im Repository!

1. Aufgabe: Implementierung Insertion Sort (bewertet)

Implementiere anhand des Pseudocodes in Listing 1 und der 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.

Mach Dich zunächst 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

```
> clang -std=c11 -Wall introprog_insertionsort.c\\
arrayio.c -o introprog_insertionsort
> ./introprog_insertionsort zahlen_insertionsort.txt
Unsortiertes Array: 72 -100 1 10 23 -1 55 9 48 2 -53 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`

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

int MAX_LAENGE = 1000;

void insertion_sort(int array[], int len) {
    /*
     * Hier Insertionsort implementieren!
     * Diese Funktion soll das Eingabearray nach dem
     * Insertionsort-Verfahren sortieren.
     * Hinweis: Verwende die "in place"-Variante! D.h. der
     * Sortiervorgang soll auf dem originalen Array stattfinden und
     * kein zweites verwendet werden.
     */
}

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

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

    /* Aufruf Insertionsort */

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

    return 0;
}
```

2. Aufgabe: Implementierung Count Sort (bewertet)

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, …, `MAX_VALUE` } sortiert werden sollen. Der Wert `MAX_VALUE` wird hierbei in der Vorgabe definiert[1].

[1] Teste Deinen Code auch mit verschiedenen `MAX_VALUE` Werten; benutze dabei im Code immer die Konstante `MAX_VALUE` anstatt einer festen Zahl wie 100.

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.

Mach Dich zunächst mit der Signatur der vorgegebenen Funktionen vertraut, um diese korrekt aufzurufen (siehe Erklärung 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

```
> clang -std=c11 -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
#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
     * Countsort einfügen!
     */

```

```
39      printf("Sortiertes Array:");
40
41      /* Folgende Zeile einkommentieren, um das Array auszugeben! */
42      // print_array(output_array, len);
43
44      return 0;
45  }
```