Fakultät IV Elektrotechnik und Informatik
Einführung in die Programmierung WS 2021/2022
Manfred Hauswirth / Damien Foucard / Christoph Seifert / Uwe Kuehn

Programmierblatt 07

Ausgabe: *16.12.2021 12:00*
Abgabe: *11.01.2022 08:00*

Thema: *Mergesort, Teile und Herrsche*

Abgabemodalitäten

1. Alle abzugebenden Quelltexte müssen ohne Warnungen und Fehler auf Deinem Rechner mit dem Befehl `clang -std=c11 -Wall -g` kompilieren.
2. Die Abgabe für den Quellcode erfolgt ausschließlich über unser Git im entsprechenden Branch. Nur wenn ein Ergebnis im ISIS-Kurs angezeigt wird, ist sichergestellt, dass die Abgabe erfolgt ist. Die Abgabe ist bestanden, wenn Du an Deinem Test einen grünen Haken siehst.
3. Du kannst bis zur Abgabefrist beliebig oft neue Versionen abgeben. Lies Dir die Hinweise der Tests genau durch, denn diese helfen Dir Deine Abgabe zu korrigieren.
Bitte beachte, dass ausschließlich die letzte Abgabe gewertet wird.
4. Die Abgabe erfolgt, sofern nicht anders angegeben, in folgendem Branch: `iprg-b<xx>-a<yy>`, wobei `<xx>` durch die zweistellige Nummer des Aufgabenblattes und `<yy>` durch die entsprechende Nummer der Aufgabe zu ersetzen sind.
5. Gib für jede Aufgabe die Quellcodedatei(en) gemäß der Vorgabe ab. Im ISIS-Kurs werden zum Teil Vorgabedateien bereitgestellt. Nutze diese zur Lösung der Aufgaben.
6. Die Abgabefristen werden vom Server überwacht. Versuche Deine Abgabe so früh wie möglich zu bearbeiten. Du minimierst so auch das Risiko, die Abgabefrist auf Grund von „technischen Schwierigkeiten“ zu versäumen. Eine Programmieraufgabe gilt als bestanden, wenn alle bewerteten Teilaufgaben bestanden sind.

Aufgabe 1 Rekursive Implementierung Mergesort (bewertet)

In dieser Aufgabe soll der **rekursive** „Teile und Herrsche“ Algorithmus *Mergesort* implementiert werden.

Implementiere die C-Funktion `merge_sort()` sowie alle nötigen Hilfsfunktionen anhand des Pseudocodes, der in der Vorlesung vorgestellt wurde. Die Funktion `merge_sort()` bekommt als Argumente die Startadresse eines Integerarrays, den Index des ersten Elements und die Länge des Arrays. Orientiere Dich am Pseudocode aus Listing 1.

Listing 1: Pseudocode Mergesort (rekursiv)

```
1  MergeSort(Array A, p, r)        // Sortiere A von index p bis r
2      if p < r then
3          q ← floor((p+r)/2)      // Mitte mit Abrunden finden
4          MergeSort(A, p, q)      // Linke Seite sortieren
5          MergeSort(A, q + 1, r) // Rechte Seite sortieren
6          Merge(A, p, q, r)       // Seiten Zusammenfuehren
7
8  Merge(A, p, q, r)
9      Array B              // Hilfsarray der Laenge r-p+1 zum Mergen
10     k ← p                // Hilfsvariable fuer linke Seite
11     m ← q + 1            // Hilfsvariable fuer rechte Seite
12     i ← 1                // Laufvariable fuer gemergtes Array
13
14     // Solange Eintraege in beiden Seiten vorhanden sind
15     while (k ≤ q) and (m ≤ r)
16         if A[k] ≤ A[m] then // Eintrag auf linker Seite kleiner oder gleich
17             B[i] ← A[k]
18             k ← k + 1
19         else                // Eintrag auf rechter Seite kleiner
20             B[i] ← A[m]
21             m ← m + 1
22         i ← i + 1           // Erhoehen der Laufvariable des gemergten Arrays
23
24     while (k ≤ q)          // Kopiere linken "Rest"
25         B[i] ← A[k]
26         k ← k + 1
27         i ← i + 1
28
29     while (m ≤ r)          // Kopiere rechten "Rest"
30         B[i] ← A[m]
31         m ← m + 1
32         i ← i + 1
33     j ← 1  // Rueckkopieren der gemergten Eintraege
34
35     while (j < i)
36         A[p + j - 1] ← B[j] // Hinweis: j ist mit 1 initialisiert
37         j ← j + 1
```

Verwende die Sortierfunktion in einem lauffähigen Programm. Das Programm bekommt als Argumente die maximale Länge des zu sortierenden Arrays und einen Dateinamen. Die Datei enthält eine Liste mit zu sortierenden Werten. Die Werte sollen mittels der vorgegebenen Funktion `read_array_from_file()` in ein Array eingelesen werden. Beachte, dass Du vor dem Einlesen entsprechend Speicher für das Array dynamisch allozieren musst.

Der Programmaufruf für ein Array mit maximal 20 Werten sieht wie folgt aus:

```
./introprog_merge_sort_rekursiv 20 zahlen_unsortiert.txt
```

Die Ausgabe des Programms sind die sortierten Werte, wobei jede Zahl durch ein Leerzeichen getrennt sein muss. Also im Format:

```
1 2 3 ...
```

Programmaufruf:

Listing 2: Programmbeispiel

```
1  > clang -std=c11 -Wall introprog_merge_sort_rekursiv.c
```

```
  input_merge_sort.c -o introprog_merge_sort_rekursiv
> ./introprog_merge_sort_rekursiv 20 zahlen_unsortiert.txt
```

Nutze zur Lösung der Aufgabe die Vorgaben aus unserem ISIS-Kurs. Füge Deine Lösung als Datei `introprog_merge_sort_rekursiv.c` im entsprechenden Abgabebranch in Dein persönliches Repository ein und übertrage die Lösung an die Abgabeplattform.

Aufgabe 2 Iterative Implementierung Mergesort (bewertet)

In dieser Aufgabe soll der **iterative** „Teile und Herrsche“ Algorithmus *Mergesort* implementiert werden. Implementiere die C-Funktion `merge_sort()` sowie alle nötigen Hilfsfunktionen anhand des Pseudocodes, der in der Vorlesung vorgestellt wurde. Die Funktion `merge_sort()` bekommt als Argumente die Startadresse eines Integerarrays, den Index des ersten Elements und die Länge des Arrays. Orientiere Dich am Pseudocode aus Listing 3.

Listing 3: Pseudocode Mergesort (iterativ)

```
IterativMergeSort(Array A, n)
    step ← 1
    while  step ≤ n do
        left ← 1
        while left ≤ n-step do
            middle ← left + step - 1
            middle ← min(middle,n)
            right ← left + 2*step - 1
            right ← min(right,n)
            merge(A, left, middle, right)
            left ← left + 2*step
        step ← step*2
```

Verwende die Sortierfunktion in einem lauffähigen Programm. Das Programm bekommt als Argumente die maximale Länge des zu sortierenden Arrays und einen Dateinamen. Die Datei enthält die Liste mit den zu sortierenden Werten. Die Werte sollten mittels der vorgegebenen Funktion `read_array_from_file()` in ein Array eingelesen werden. Beachte, dass Du vor dem Einlesen entsprechend Speicher für das Array dynamisch allozieren musst.

Das Programm soll für ein Array mit maximal 20 Einträgen wie folgt aufgerufen werden:

```
./introprog_merge_sort_iterativ 20 zahlen_unsortiert.txt
```

Ausgabe des Programms sind die sortierten Werte, wobei jede Zahl durch ein Leerzeichen getrennt sein muss. Also im Format:

```
1 2 3 ...
```

Programmaufruf:

Listing 4: Programmbeispiel

```
> clang -std=c11 -Wall introprog_merge_sort_iterativ.c
  input_merge_sort.c -o introprog_merge_sort_iterativ
> ./introprog_merge_sort_iterativ 20 zahlen_unsortiert.txt
```

Nutze zur Lösung der Aufgabe die Vorgaben aus unserem ISIS-Kurs. Füge Deine Lösung als Datei `introprog_merge_sort_iterativ.c` im entsprechenden Abgabebranch in Dein persönliches Repository ein und übertrage die Lösung an die Abgabeplattform.