MERGESORT

INDINF04

23. OKTOBER 2014

STEFAN ERCEG
4AHITT

Inhalt

Aufgabenstellung

Es soll ein C-Programm geschrieben werden, welches den Sortieralgorithmus "Mergesort" implementiert. Bei der Implementierung sollen auf jeden Fall Pointer verwendet werden.

Als Input soll ein randomized gefülltes Array in einem Struct verwendet werden (siehe Tipps). Die Ausgabe des unsortierten und dann sortierten Arrays soll mittels einer eigenen Funktion erfolgen, die über einen Functionpointer aus der Struct erreichbar sein soll (Übergabe mittels call-by-reference).

Zeitabschätzung

Teilaufgabe	**Benötigte Zeit**
Code erstellen	80 Minuten
Protokoll schreiben	60 Minuten
Gesamt	**140 Minuten (2 h 20 min)**

Tatsächlicher Zeitaufwand

Teilaufgabe	Datum	Benötigte Zeit
Code erstellen	14.10.2014	35 Minuten
Code erstellen	22.10.2014	30 Minuten
Protokoll schreiben	23.10.2014	45 Minuten
Gesamt	23.10.2014	**110 Minuten (1h 50 min)**

Designüberlegung

Struct

struct randomizedArray

Es wird eine Struct erstellt, welche ein int-Array und zwei Functionpointer beinhaltet.

Funktionen

int* zufallszahlen(int* zahlen)

Die Funktion „zufallszahlen“ generiert 10 Zufallszahlen und gibt diese als int-Array zurück. Dazu wird ein int-Array als Parameter übergeben.

void mergesort(int liste[], int groesse)

Die Funktion „mergesort“ wurde von folgender Seite unverändert übernommen: http://de.wikibooks.org/wiki/Algorithmen_und_Datenstrukturen_in_C/_Mergesort. Zuletzt wurde diese Seite am 22.10.2014 um 22 Uhr abgerufen.
Als Parameter werden das zu sortierende Array und dessen Länge übergeben.

Allgemeines zum Mergesort

Beim Mergesort werden die Zahlen in einer Liste betrachtet und in kleinere Listen zerlegt, die dann jeweils für sich sortiert werden. Die sortierten kleinen Listen werden danach wieder zu einer Gesamtliste hinzugefügt.

void output(int* zahlen)

Die Funktion „output“ gibt alle Zahlen, die im Array enthalten sind, aus. Die Zahlen werden durch Beistriche getrennt.

Things I've done

Functionpointer

In der Struct habe ich zwei Functionpointer folgendermaßen deklariert:

```
void (*sort) (int*, int);
void (*output) (int*);
```

Bei der Definition eines Functionpointers legt man daher fest, wie die Funktion in Bezug auf Rückgabetyp, Datentypen der Parameter, etc. aussieht.

Damit der Funktionspointer „sort" auf die Funktion „mergesort" zeigt, führt man eine einfache Zuweisung aus:

```
struct randomizedArray ra;
ra.sort = mergesort;
```

Zahlen ausgeben

```
void output(int* zahlen) {
	for (int i = 0; i < MAX; i++)	{
		if (i == MAX-1) {
			printf("%d", zahlen[i]);
		} else {
			printf("%d, \t ", zahlen[i]);
		}
	}
	printf("\n");

}
```

Da ich die Zahlen durch Beistriche getrennt anzeigen wollte, musste ich eine if-Anweisung durchführen, welche überprüft, ob man beim letzten Element des Arrays angekommen ist. Wenn dies nämlich geschieht, soll kein Beistrich nach der Zahl hinzugefügt werden.

Lessons learned

- ➔ Umgang mit Structs
- ➔ Umgang mit Functionpointern
- ➔ Structs fassen mehrere Variablen (= Komponenten) zu einer Struktur zusammen. Die Variablen können verschiedene Datentypen besitzen.
- ➔ Functionpointer sind praktisch, da sie nicht auf einen Bereich im Datensegment, sondern auf eine Funktion im Codesegment zeigen. Der Vorteil ist der, dass man zur Laufzeit entscheiden kann, welche Funktion aufgerufen werden soll und dies eine Flexibilität gewährleistet.

Quellenangaben

Funktion „mergesort“:
http://de.wikibooks.org/wiki/Algorithmen_und_Datenstrukturen_in_C/_Mergesort