2. Oktober 2014

Team: 02, Daniel Hofmeister, Alex Mantel

Aufgabenaufteilung:

1. Daniel Hofmeister: Skizze, Selection Sort
2. Alex Mantel:

Quellenangaben:

- Praktikumsaufgabe 2
- Erlang Utility

Bearbeitungszeitraum:

- Daniel Hofmeister: 4h
- Alex Mantel:

Aktueller Stand: Selection Sort implementiert.

Fragen an: **alex.mantel@haw-hamburg.de**
daniel.hofmeister@haw-hamburg.de

Änderungen der Skizze:

2. Oktober 2014

Aufgabe 3: Implementierung von einfachen Suchalgorithmen

1. sortNum(Anzahl,Case):

[Case] rand: Es soll eine Datei (zahlen.dat) erstellt werden, die aus [Anzahl] Zufallszahlen besteht um die kommenden Suchalgorithmen zu testen. Hierfür sollte der Aufruf „zahlenfolge/4“ aus der Util.erl Datei verwendet werden. Das „Minimum“ und „Maximum“ sollte natürlich sinnvoll belegt werden.

[Case] BC (best case): Es soll eine Datei (zahlen.dat) erstellt werden, die aus [Anzahl] Zahlen sortier Zahlen von links nach rechts aufsteigend besteht. Zur Erzeugung der Liste biete sich hier sortliste/1 aus der Util.erl Datei an.

[Case] WC (worst case): Es soll eine Datei (zahlen.dat) erstellt werden, die aus [Anzahl] Zahlen sortier Zahlen von links nach rechts aufsteigend besteht. Zur Erzeugung der Liste biete sich hier resortliste/1 aus der Util.erl Datei an.

2. `insertionSort(Array):`

Ein Array aus Zahlen soll mit diesem Aufruf sortiert werden, aufsteigend von links nach rechts. Der verwendet Algorithmus soll hierbei der insertionSort und die Lösung iterativ-rekursiv gestaltet sein. Dabei sollte man das Array in zwei Teile aufspalten, ein „sorted Array“ und ein „unsorted Array“. Das erste Element des unsorted Array sollte dabei bei jedem Aufruf genommen werden und an die korrekte Stelle im sorted Array gesteckt werden.

3. selection`Sort(Array):`

Ein Array aus Zahlen soll mit diesem Aufruf sortiert werden, aufsteigend von links nach rechts. Der verwendet Algorithmus soll hierbei der selectionSort und die Lösung iterativ-rekursiv gestaltet sein. Hierbei soll das jeweils noch nicht sortierte, niedrigste Element gesucht und an die erste nicht sortierte Stelle geschrieben werden.

Für beide Algorithmen (2+3) gilt:

Das jeweils zu sortierende Array soll aus einer Datei (zahlen.dat) ausgelesen werden, das nach Anwendung des Algorithmus sortierte Array soll selbst in eine Datei (sortiert.dat) ausgegeben werden.

Zusätzlich soll eine Zeitmessung eingebaut werden, um die Dauer einer einzelnen Sortierung zu bestimmen. Eine zweite Kopie des codes soll am Ende erstellt werden und diese mit Zählern für Vergleiche und Verschiebung der Elemente versehen.

Weiterhin soll es möglich sein, die Algorithmen 3 Stellig aufzurufen (Array,Pos1,Pos2) und nur das Teilarray zwischen Pos1 und Pos2 zu sortieren.

2. Oktober 2014

Abschließend:

Die Algorithmen sind 100-mal auszuführen, wenn möglich mit jeweils unterschiedlichen Zahlen: 80-mal Zufallszahlen und jeweils 10-mal "best case" bzw. "worst case" Zahlen. Aus den Zeitmessungen ist dann per Mittelwert anzugeben:

1. Anzahl eingelesener Elemente
2. Name des Algorithmus
3. Benötigte Zeit, sowie maximal und minimal benötigteZeit.
4. Anzahl Vergleiche, sowie maximale Anzahl und minimale Anzahl an Vergleichen.
5. Anzahl Verschiebungen, sowie maximale Anzahl und minimale Anzahl an Verschiebungen.

Die Daten sind in einer Datei `messung.log` zu speichern.