

SWE3_Deutz_Ue02

DAVID DEUTZ

Table of Contents

Beispiel 1: External Merge Sort	2
Lösungsidee	2
Lösungsidee File Manipulator	2
Lösungsidee External Merge Sort.....	2
Code.....	3
Testfälle	3
Testfälle File Manipulator.....	3

Beispiel 1: External Merge Sort

Lösungsidee

Lösungsidee File Manipulator

Der File-Manipulator dient für verschiedene Operationen, welche auf Files ausgeführt werden können, darunter zum Beispiel das Kopieren, das zufällige Befüllen mit (in meinem Fall) Zahlen oder das Aufteilen von Dateien.

Das zufällige Befüllen einer Datei wird mit der Funktion „FillRandomly“ bereitgestellt. Dieser werden ein Maximum und ein Minimum sowie die Anzahl der Zahlen, die in der Datei stehen sollen, genannt. Anschließend befüllt die Funktion eine Datei mit zufälligen Zahlen zwischen dem Minimum und dem Maximum so lange, bis die gewünschte Anzahl erreicht ist.

Das Aufsplitten der Dateien funktioniert mit der Funktion „SplitFile“. Hierbei benötigt die Funktion den Namen der Datei, welche aufgeteilt werden soll und eine Reihe von Filenamen, in welchen das Ergebnis der Aufteilung stehen soll. Die Funktion liest anschließend Zahl für Zahl von der Ursprungsdatei ein und schreibt sie abwechselnd in die Zielformateien.

Des Weiteren kann der File-Manipulator mit der Funktion „CopyFile“ eine Datei kopieren. Sie liest von einem ursprünglichen File die Values ein und schreibt sie im selben Schritt auf die neue Datei aus. Sie benötigt nur die Namen der Ursprungsdatei und den der Datei, in die kopiert werden soll.

Mithilfe des File-Manipulators kann man auch eine Reihe von Werten von einer Datei einlesen oder auf eine Datei ausschreiben mithilfe der beiden Funktionen „ReadFromFile“ und „WriteToFile“.

Lösungsidee External Merge Sort

Die Funktion „Sort“ arbeitet unter anderem mithilfe des File-Manipulators. Die zu sortierende Datei wird im ersten Schritt in zwei gleich große Files geteilt. Anschließend wird mit einem Lauf der Länge 1 begonnen zu sortieren, was bedeutet, dass immer ein Wert aus jeder Datei eingelesen und gespeichert wird. Anschließend werden die gespeicherten Werte mithilfe der Funktion „merge“ sortiert und verbunden und in eine andere Datei gespeichert. Dann werden die nächsten Werte eingelesen, so lange bis alle Werte der Datei eingelesen wurden. Anschließend wird die Länge des Laufs verdoppelt und die Werte kommen nun von den in den neuen Dateien gespeicherten Werten. Diese werden wieder eingelesen, sortiert und nun in die ursprüngliche Datei gespeichert. Das bedeutet, es existieren vier Dateien, die abwechselnd dazu dienen Werte zu speichern und dann einzulesen. Sobald alle Werte sortiert wurden, wird das Ergebnis in einer Enddatei gespeichert, welche nun eine sortierte Version der Anfangsdatei darstellt.

Die Funktion „Merge“ verbindet gespeicherte Werte in einer sortierten Form. Dazu werden zwei Indizes verwendet, welche über jeweils eine der beiden Listen, in denen die Werte gespeichert sind, gehen. Die Werte an den beiden Indizes werden miteinander verglichen und der kleinere

abgespeichert. Anschließend wird der Index des abgespeicherten Wertes erhöht und mit dem anderen Wert verglichen. Sollte ein Index am Ende der Liste angekommen sein, kann automatisch die restliche andere Liste gespeichert werden, da die beiden Listen bereits sortiert sein müssen.

Code

Siehe Visual Studio

Testfälle

Testfälle File Manipulator

Test „FillRandomly“: file mit 100 zufälligen Werten zwischen 0 und 100 füllen

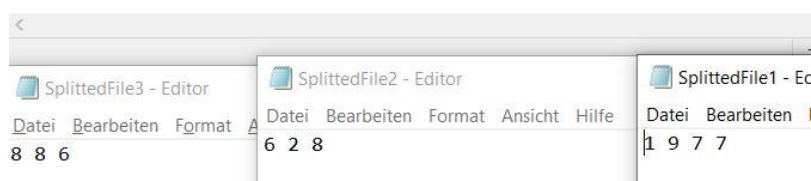
Der Screenshot ist nur ein kleiner Ausschnitt aus dem entstandenen File

```
63 79 97 81 24 32 99 39 8 89 27 78 67
```

Test „SplitFile“: ein File 10 random werten aufsplitten auf 3 files



```
Datei Bearbeiten Format Ansicht Hilfe
1 6 8 9 2 8 7 8 6 7
```




```
SplittedFile3 - Editor
Datei Bearbeiten Format Ansicht Hilfe
8 8 6

SplittedFile2 - Editor
Datei Bearbeiten Format Ansicht Hilfe
6 2 8

SplittedFile1 - Editor
Datei Bearbeiten
1 9 7 7
```

Test „Copy“: das File „fill_random_test.txt“ in „fill_random_test_copy.txt“ kopieren

```
63 79 97 81 24 32 99 39 8 89 27 78 67 72 5 32 0 72 14 35 31 42 58 77 88 68 89 67 89 47 96 58 :
```



```
fill_random_test_copy - Editor
Datei Bearbeiten Format Ansicht Hilfe
63 79 97 81 24 32 99 39 8 89 27 78 67 72 5 32 0 72 14 35 31 42 58 77 88 68 89 67 89 47 96 58
```

Aus zetilichen Gründen konnte ich die Funktion „Sort“ nicht fertig implementieren und auch die Tests sind weniger. Tests, die ich mit der fertigen Sort Funktion gemacht hätte wären jedoch:

- **Leeres File**
- **Mit einem Wert**
- **Mit mehreren Werten**
- **Je nach zeit noch mit drei verschiedenen input und output files**
- **Große menge an zahlen**