

## Übungsblatt 2

Ziel dieses Übungsblattes ist es, dass Sie sich mit den Funktionen und dem Prinzip der Dateiverarbeitung vertraut machen.

### Aufgabe 1 – Lesen und Verarbeiten von Textdateien

Schreiben Sie eine Funktion *int countLetters (FILE\* fin)*; die die Anzahl der Buchstaben in einer Textdatei bestimmt. Testen Sie Ihre Funktion aus, indem Sie Ihre Funktion mit folgendem Hauptprogramm für die Datei count.txt austesten.

```
int countLetters(FILE* fin);

int main (void)
{
    int numberOfLetters;
    char fileName[256];

    FILE* fp;
    printf("Geben Sie den Namen der Datei ein: ");
    scanf("%s", fileName);

    /*Hier ergänzen */

    printf("Die Anzahl an Buchstaben in der Datei| %s ist: %d", fileName, numberOfLetters);
    fclose(fp);
    return 0;
}
```

### Aufgabe 2 – Erzeugen und Schreiben von Textdateien

Schreiben Sie ein Programm, das

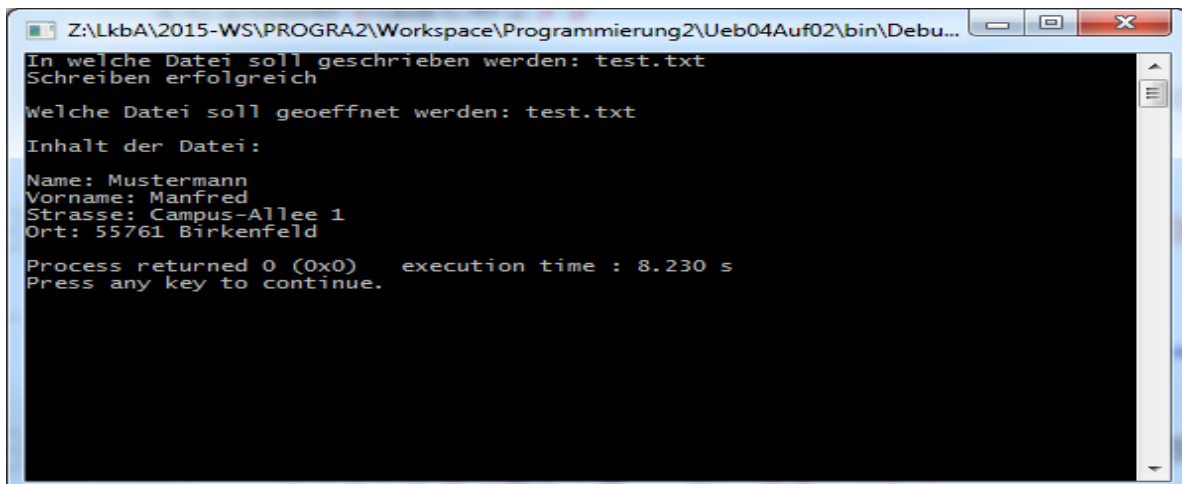
*Name: Mustermann*

*Vorname: Manfred*

*Strasse: Campus-Allee 1*

*Ort: 55761 Birkenfeld*

in eine Datei schreibt. Der Name der Datei soll durch das Programm abgefragt werden. Anschließend soll die Datei geöffnet werden. Die Ausgabe soll über die Funktion `fprintf()` erfolgen. Die Konsolenausgabe soll wie folgt aussehen:



### Aufgabe 3 – Einlesen und Sortieren von Zahlen

Schreiben Sie ein Programm, das ganze Zahlen aus einer Textdatei sort.txt ausliest und sortiert auf dem Bildschirm ausgibt. Die Datei ist so aufgebaut, dass die erste Zahl die Anzahl der zu sortierenden Zahlen angibt. Der Name der Datei soll im Hauptprogramm, wo die Datei auch geöffnet werden soll, abgefragt werden. Verwenden Sie zum Einlesen die Funktion `fscanf()` und zum Schreiben die Funktion `fprintf()` auf dem Standard-Ausgabepuffer `stdout`.

Schreiben Sie eine Funktion **`int* readNumbers(FILE *fin, int *n)`**, die die Zahlen aus der Datei, auf die `fin` zeigt, ausliest und in einem dynamischen Feld speichert. Das Feld wird innerhalb der Funktion mit `malloc()` angelegt. Die Funktion liefert einen Zeiger auf das Feld zurück. Unter der in `n` übergebenen Adresse wird die Anzahl der gelesenen Zahlen gespeichert. Gehen Sie davon aus, dass die Datei korrekt geöffnet wurde und dass das Einlesen der Zahlen fehlerfrei erfolgt.

```
int main(void) {  
    int n, *p;  
    FILE *infp;  
    if((infp = fopen ("sort.txt", "r")) == NULL) {  
        printf("Die Datei sort.txt konnte nicht geoeffnet werden\n");  
        exit(1);  
    }  
    p = readNumbers(infp, &n);  
    sort(p, n);  
    //Ausgabe der Zahlen auf dem Bildschirm  
    free(p);  
    fclose(infp);  
    return 0;  
}
```

## Aufgabe 4 – Erzeugen von Strukturen aus Textdateien

Laden Sie sich die Datei item.txt aus dem Stud.IP und fügen Sie diese Datei ihrem Spiele-Projekt hinzu. In dieser Datei sind Gegenstände beschrieben.

```
10
Armbrust
3,3.5,25,0

Heilamulett
1,0.25,100,2

Eisenhelm
0,1.5,35,0
```

Die Datei ist folgendermaßen aufgebaut. In der 1. Zeile steht die Anzahl der Gegenstände in der Datei. Danach folgen die Gegenstände. Jeder Gegenstand ist über zwei Zeilen beschrieben. In der ersten Zeile steht der Name des Gegenstandes in der folgenden Zeile stehen die Attribute des Gegenstandes (Position, Gewicht, Wert und LP\_Change) jeweils durch Komma getrennt. Danach kommt eine Leerzeile bevor der nächste Gegenstand folgt.

Erstellen Sie zunächst eine globale Pointer-Variable für die Gegenstände und nennen Sie diese treasures.

```
//{ Globalen Variablen
Item NOTHING = { "Leer", head, 0, 0 };
Room WALL = { "Wand", NULL, 0, NULL, NULL };
Item* treasures;
```

Anschließend implementieren Sie eine Funktion

```
void readTreasureFile(char*);
```

Diese Funktion bekommt den Namen der Datei übergeben. Liest anschließend die Datei ein und allokiert dynamisch so viel Speicherplatz wie Gegenstände in der Datei stehen.

Anschließend werden die Gegenstände erzeugt und in der globalen Variable gespeichert.

Überprüfen Sie ob alles funktioniert, indem Sie abschließend mit einer Schleife über das treasure-Array laufen und die Gegenstände mit Hilfe der printItem-Funktion ausgeben.