

Hardwarenahe Programmierung
Gruppe 11 (Thomas)

In dieser Übung arbeiten Sie mit komplexen Datentypen in C-Programmen, die über structs realisiert werden.

- *Denken Sie wie immer daran, Ihre Lösungen im ILIAS hochzuladen und den Test im ILIAS zu absolvieren!*
- *Sie brauchen keine ungültigen Eingaben oder semantisch falschen Daten abzufangen, wie zum Beispiel das Datum 2020-13-40.*
- *Die Inhalte aller Übungen werden relevant für das Abschlussprojekt sein. Versuchen Sie trotzdem, alle Aufgaben zu lösen, auch wenn Sie die minimal nötige Anzahl an bestandenen Aufgaben bereits erreicht haben!*

Aufgabe 1 *Haustier (Pflichtaufgabe)*

Ergänzen Sie die Dateien `haustier.h` und `haustier.c`, sodass die gegebenen Unittests erfüllt werden.

Ein Haustier wird repräsentiert durch ein `struct` und hat einen Namen, der höchstens 10 darstellbare Zeichen lang ist. Implementieren Sie weiterhin

- eine Funktion `name_setzen`, die den Namen des Haustiers setzt,
- eine Funktion `name_ausgeben`, die den Namen in ein übergebenes Array schreibt,
- eine Funktion `name_umdrehen`, die dem Namen des Tiers umdreht.

Verwenden Sie den Befehl `make run`, um Ihre Implementierung zu testen.

Aufgabe 2 *Datenstrukturen (Pflichtaufgabe)*

- (a) Erstellen Sie eine Headerdatei `datum.h`, in welcher Sie ein `struct datum` definieren, das ein Datum mit Tag, Monat und Jahr speichern kann.
- (b) Fügen Sie Ihrer Headerdatei einen Prototypen für eine Funktion hinzu, die ein Datum über einen Zeiger mit Jahr, Monat und Tag initialisiert:

```
void datum_setzen(struct datum *d, int jahr, int monat, int tag);
```

Implementieren Sie diese Funktion in der Datei `datum.c`.

- (c) Ergänzen Sie Ihre Dateien um eine weitere Funktion, die ein Datum in einen Datenstrom schreibt.

```
void datum_ausgeben(FILE *ausgabe, struct datum *d);
```

Verwenden Sie hierbei das internationale Datumsformat `JJJJ-MM-TT` (Beispiel: `2020-01-30`). In diesem Format werden zuerst das Jahr, dann der Monat und schließlich der Tag ausgegeben. Achten Sie auf die Ausgabe führender Nullen. Hierbei hilft es, die Dokumentation bekannter Ausgabefunktionen zu lesen.

Die Datei wird von der aufrufenden Funktion geöffnet und geschlossen. Dateien dürfen nicht doppelt geschlossen werden, denn dies würde zu einem Fehler führen. Sie dürfen die Datei also nicht selber öffnen oder schließen.

Schreiben Sie nach dem Datum keinen Zeilenumbruch in die Datei.

- (d) Ergänzen Sie Ihre Dateien um eine weitere Funktion, die zwei Datum-Objekte vergleicht:

```
int datum_vergleichen(struct datum *a, struct datum *b);
```

Diese Funktion soll folgende Werte zurückgeben:

- 0, wenn beide eingegebenen Daten gleich sind;
- einen beliebigen negativen Wert, wenn `a` zeitlich vor `b` liegt;
- einen beliebigen positiven Wert, wenn `a` zeitlich nach `b` liegt.

- (e) Testen Sie Ihre Funktionen mit dem Befehl `make run`.

Aufgabe 3 Supermarkt (*Pflichtaufgabe*)

Ein Supermarkt möchte sein Inventar verwalten. Dies ist aufgeteilt in drei Dateien:

- Verfallsdatum (im Format `Jahr Monat Tag`, Eingabe ohne und Ausgabe mit Bindestrichen)
- Kategorie (entweder `Backware`, `Gemuese`, `Getraenk` oder `Obst`)
- Bezeichnung (höchstens 10 Zeichen)

Jede Zeile der drei Dateien entspricht dabei einem Produkt. Ihre Aufgabe ist es, die Daten aus den drei Dateien zusammenzuführen und nach Verfallsdatum gefiltert in eine Datei auszugeben.

Das Resultat sieht dann beispielsweise so aus:

2022-01-09 Pflaume Obst

2022-08-12 Kartoffel Gemuese

- Implementieren Sie in `supermarkt.h` ein `enum kategorie`, dass die Produktkategorie speichert.
- Implementieren Sie in `supermarkt.h` ein `struct produkt`, dass das Verfallsdatum, die Kategorie und die Bezeichnung speichert. Für das Verfallsdatum **muss** das `struct` aus `datum.h` der vorherigen Aufgabe wiederverwendet werden. Zur Speicherung der Produktkategorie **muss** das vorher erstellte `enum` verwendet werden. Wenn Sie andere als diese drei Komponenten in ihrem `struct` deklarieren, dann wird diese Aufgabe als nicht bestanden bewertet.
- Implementieren Sie in `supermarkt.c` die Funktion `produkte_einlesen`, die die Daten aus drei gegebenen Dateien einliest, in ein Array von Produkten schreibt und die Anzahl zurückgibt.
- Implementieren Sie in `supermarkt.c` die Funktion `produkte_ausgeben`, die Produkte aus einem Array in eine Datei schreibt, wenn das gegebene Verfallsdatum noch nicht überschritten wurde.
- Sehen Sie sich zum besseren Verständnis der Aufgabenstellung die Dateien `supermarkt.c` und `supermarkt.h` an.
- Die `main`-Funktion darf nicht verändert werden!
- Testen Sie Ihr Programm mit dem Befehl `make run`.