Institut für Informatik
Prof. Dr. Stefan Conrad

Sommersemester 2020
Übung 5

Hardwarenahe Programmierung
Gruppe 05 (Florian)

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 semantisch falschen Daten abzufangen, wie zum Beispiel das Datum `2020-13-40`.*

Aufgabe 1 *Datenstrukturen (**datum.c** und **datum.h** abgeben.)*

(a) Erstellen Sie eine Headerdatei **datum.h**, in welcher Sie eine Datenstruktur (`struct`) definieren, die ein Datum mit Tag, Monat und Jahr speichern kann. Nutzen Sie Ihre Datenstruktur, um einen Datentypen mit Namen `datum` zu definieren.

(b) Fügen Sie Ihrer Headerdatei einen Prototypen für eine Funktion hinzu, die das Jahr, den Monat und den Tag entgegen nimmt und daraus ein Datumsobjekt erzeugt:

```
datum neues_datum(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-Objekt entgegen nimmt und in einen Datenstrom schreibt.

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

Verwenden Sie hierbei das internationale Datumsformat `JJJJ-MM-TT` (Beispiel: `2020-12-24`). In diesem Format werden zuerst das Jahr, dann der Monat und schließlich der Tag ausgegeben.

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

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

Diese Funktion soll folgende Werte zurückgeben:

- 0, wenn beide eingegebenen Daten gleich sind;
- -1, wenn `a` zeitlich vor `b` liegt;
- 1, wenn `a` zeitlich nach `b` liegt.

(e) Testen Sie Ihre Funktionen mit den vorgegebenen Unit-Tests (**make run**).

Aufgabe 2 *Auto-Daten (**autos.c** und **autos.h** abgeben.)*

Ein reicher Autosammler möchte die folgenden Daten seiner Autos digital verwalten:

- Herstellername (weniger als 100 Zeichen),
- Modell (weniger als 100 Zeichen),
- Baujahr,
- Klimaanlage (ja/nein, weniger als 100 Zeichen),
- Erwerbsdatum (Tag, Monat und Jahr)
- und Fahrzeugklasse (Kombi, Limousine, Minivan oder Kleinwagen).

Die Autos sind gespeichert in der Datei `autos_eingabe.txt`, deren Inhalt Beispielsweise so aussieht:

```
Hersteller2 M0816 2015 nein 1 5 2015 Limousine
Hersteller1 M0817 2015 ja 1 4 2015 Minivan
```

Dazu erhalten Sie noch weitere Dateien.

(a) Implementieren Sie in `autos.h` ein `enum`, dass die Fahrzeugklasse speichert. Überlegen Sie sich anhand der gegebenen Dateien, welche Fahrzeugklassen dazu nötig sind.

(b) Implementieren Sie in `autos.h` ein `struct`, dass die Daten des Fahrzeugs speichert. Überlegen Sie sich, welche Komponenten von welchem Datentyp dazu nötig sind. Sie dürfen hierzu die Dateien `datum.c` und `datum.h` der vorherigen Aufgabe wiederverwenden.

(c) Implementieren Sie in `autos.c` eine Funktion, die ein Auto einliest.

```
void fahrzeug_einlesen(fahrzeug *einauto, FILE *eingabe)
```

(d) Implementieren Sie in `autos.c` eine Funktion, die das Auto mit dem neusten Erwerbsdatum findet und als Pointer zurück gibt.

```
fahrzeug* neuestes_fahrzeug(fahrzeug *autos, int anzahl)
```

In der Datei `main.c` werden Ihre Funktionen dann aufgerufen, um die Autos einzulesen, auszugeben und das neuste Auto zu finden.

(e) Testen Sie Ihr Programm mit dem Test-Skript `test.sh`.

Aufgabe 3 *Brüche (**bruch.c** und **bruch.h** abgeben)*

Ergänzen Sie die gegebenen Dateien *bruch.h* und *bruch.c* so, dass sie die gegebenen Unittests erfüllen. Sie können die Datei *main.c* verwenden, um Ihre Implementierung auszuprobieren.

Die Dateien sollen:

- eine Datenstruktur `bruch` zur Speicherung eines Bruchs enthalten,
- eine Funktion `product` enthalten, die das Produkt zweier Brüche zurückgibt,
- eine Funktion `set_nenner` enthalten, die den Nenner eines gegebenen Bruchs auf einen neuen, gegebenen Wert ändern kann,
- und eine Funktion `print` enthalten, die den Bruch als Gleitkommazahl auf der Konsole ausgibt.

Falls der Nenner eines Bruchs 0 werden sollte, setzen Sie ihn auf 1, um nicht durch 0 zu teilen und das Universum nicht zu zerstören.

Aufgabe 4 *Platzsparende Datenstrukturen*

Gesucht ist eine Datenstruktur, in der Antworten eines Feedbackfragebogens zu einer Übung gespeichert werden können. Implementieren Sie diese Datenstruktur und verwenden Sie dabei Enums und Bitfelder, um möglichst wenig Speicher zu verwenden. *Bonus*: Verwenden Sie zusätzlich dynamische Speicherverwaltung, um möglichst wenig Platz für die Speicherung von Strings zu belegen.

Der Fragebogen beinhaltet die folgenden Fragen (Antwortmöglichkeiten in Klammern):

1. "Welches Fach studieren Sie?" (Informatik/Mathematik/Physik)
2. "In welchem Fachsemester studieren Sie?" (1-12)
3. "Bitte benoten Sie Ihre Übung in Schulnoten." (1-6)
4. "Haben Sie weitere Anmerkungen?"
   (Freitext, maximal 120 Zeichen)

Betten Sie Ihre Datenstruktur in ein Programm ein, welches einen solchen Fragebogen vom Benutzer abfragt, die Antworten speichert und wieder auf der Konsole ausgibt. Wir haben Ihnen einen Prototypen bereitgestellt, welcher die Eingabe schon vorbereitet.