Institut für Informatik
Prof. Dr. Stefan Conrad

Sommersemester 2020
Übung 5

Hardwarenahe Programmierung
Gruppe 14 (Lars)

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`.

Aufgabe 1 *Brüche (**Pflichtaufgabe**)*

Ergänzen Sie die gegebenen Dateien `bruch.h` und `bruch.c` so, dass sie die gegebenen Unittests erfüllen.

Die Dateien sollen:

- eine Datenstruktur `bruch` zur Speicherung eines Bruchs enthalten,
- eine Funktion `product` enthalten, die zwei Brüche multipliziert und das Produkt zurückgibt,
- eine Funktion `kuerze` enthalten, die einen Bruch soweit wie möglich kürzt,
- und eine Funktion `print` enthalten, die einen Bruch als Gleitkommazahl in ein char-Array schreibt.

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 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: `2020-01-30`). 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 keine anderen Zeichen in die Datei.

(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;
- 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 *Studenten-Daten (**Pflichtaufgabe**)*

Für die Verwaltung einer Vorlesung sollen folgende Studentendaten verarbeitet werden:

- Name (weniger als 100 Zeichen),
- Vorname (weniger als 100 Zeichen),
- Matrikelnummer,
- Geburtsdatum (Tag, Monat, Jahr),
- Klausurzulassung (ja/nein)
- und Fach (Informatik, Physik oder Mathematik).

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

```
Maus Micky 12345 18 11 1928 ja Mathematik
Duck Donald 23456 13 3 1920 ja Informatik
```

Dazu erhalten Sie noch weitere Dateien.

(a) Implementieren Sie in `studenten.h` ein `enum`, dass das Studienfach speichert. Überlegen Sie sich anhand der gegebenen Dateien, welche Fächer dazu nötig sind.

(b) Implementieren Sie in `studenten.h` ein `struct`, dass die Daten eines Studenten speichert. Überlegen Sie sich, welche Komponenten von welchem Datentyp dazu nötig sind. Verwenden Sie hierzu die Dateien `datum.c` und `datum.h` der vorherigen Aufgabe wieder.

(c) Implementieren Sie in `studenten.c` eine Funktion, die einen Studenten einliest.

```
void student_einlesen(student *einstudent, FILE *eingabe)
```

Funktionen zum Einlesen und Ausgeben des Studienfachs sind für Sie bereits implementiert.

(d) Implementieren Sie in `studenten.c` eine Funktion, die den ältesten Studenten findet und als Pointer zurück gibt.

```
student* aeltester_student(student *studenten, int anzahl)
```

In der Datei `main.c` werden Ihre Funktionen dann aufgerufen, um die Studenten einzulesen, auszugeben und den ältesten Studenten zu finden und ebenfalls auszugeben.

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

Aufgabe 4 *Platzsparende Datenstrukturen (Vorbereitungsaufgabe)*

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.