Zusatzaufgabe: Notendurchschnitt

Im Rahmen Ihres Studiums an der HAW müssen Sie diverse Prüfungen ablegen, deren Ergebnisse die Note Ihres Bachelor-Abschlusses beeinflussen. Die Abschlussnote n_a berechnet sich aus dem gewichteten Mittelwert aller Ihrer im Studium erzielten Noten. Das hier zu erstellende Programm soll diese Abschlussnote berechnen. Grundlage dafür stellt die Prüfungsordnung für den REE-Studiengang mit Stand 30.1.2020 dar[1].

Zur Abschlussnote tragen $N = 33$ Module mit den Noten $n_1..n_N$ und den dazugehörigen Gewichtungsfaktoren $g_1..g_N$ bei. (Neben den 33 benoteten Modulen gibt es noch weitere Module ohne Noten bzw. Gewichtung, die hier aber nicht berücksichtigt werden.) Es ergibt sich folgende Formel:

$$n_a = \frac{\sum_{i=1}^{N} g_i \cdot n_i}{\sum_{i=1}^{N} n_i}$$

Das zu erstellende Programm soll möglichst komfortabel alle Noten des Studiengangs vom User abfragen und daraus die Abschlussnote ermitteln.

Schritt 0:

In EMIL finden Sie eine sogenannte CSV[2]-Datei, in der alle Module gelistet sind. Die erste Zeile dient als Überschrift. Danach folgt in jeder Zeile ein Modul mit der Modulbezeichnung, Kürzel, Gewichtung und einer Note. Laden Sie diese Datei herunter und machen Sie sich mit der Dateistruktur und den Daten vertraut.

Schritt 1: Datenstruktur & Datentyp anlegen

Definieren Sie eine **Struktur** für ein Fach mit den Elementen *Modulbezeichnung*, *Abkuerzung*, *Gewichtung* und *Note*. (Die Namen können Sie auch anders vergeben.) Für die Modulbezeichnung und die Abkürzung verwenden Sie eine konstante Zeichenkettenlänge. Definieren Sie für diese Struktur einen **neuen Datentyp**. Definieren Sie im Hauptprogramm einen **Zeiger** auf diese Struktur.

Schritt 2: Daten einlesen

Lesen Sie aus der gegebenen CSV-Datei die Daten für die unterschiedlichen Module ein, nutzen Sie hierfür die Struktur und den Zeiger aus Schritt 1 und allokieren Sie dynamisch Speicher für jedes neue Modul, welches Sie aus der Datei auslesen. Ihr Code soll mit einer beliebigen Anzahl von Modulen (d.h. beliebige Anzahl von Zeilen in der CSV-Datei) funktionieren.

Tipp: Das Einlesen der Daten aus der Datei kann z.B. zeilenweise mit der Funktion `fgets()` erfolgen. Jede Zeile wird zwischengespeichert und dann verarbeitet. Zur Verarbeitung bietet sich die Funktion `strtok()` an, welche einen String anhand bestimmter Trennzeichen zerlegt. Weitere Details finden Sie in einer Codereferenz[3].

[1] https://www.haw-hamburg.de/studium/studienorganisation/ordnungen/pruefungs-und-studienordnungen/

[2] CSV steht für Comma-Separated Values und ist ein typisches Dateiformat, um Daten auszutauschen. Es ist eine Textdatei, um Tabellen oder Listen strukturiert darzustellen. Die Daten in einer Zeile werden durch ein festgelegtes Zeichen (z.B. Komma oder Semikolon) voneinander getrennt. Häufig sind dies verschiedene Werte für ein Datensatz. Darüber hinaus wird der Zeilenumbruch genutzt, um auch mehrere Datensätze voneinander zu trennen.

[3] Z.B. https://cplusplus.com/reference/cstring/strtok/

Schritt 3: Ausgabe

Erstellen Sie eine Funktion, die alle Module mit Abkürzung und Note auf dem Bildschirm ausgibt.

Schritt 4: Noten eingeben

Erstellen Sie eine Funktion, mit der Sie für alle Fächer die Noten vom Benutzer abfragen. Die Eingabe soll auf Gültigkeit (Werte zw. 0 bis 15) überprüft werden.

Schritt 5: Berechnung der Abschlussnote

Erstellen Sie eine Funktion zur Berechnung und Ausgabe der Abschlussnote. Es soll

- die Summe aller Punkte, d.h. der Zähler der in der Einleitung genannten Formel,
- die Abschlussnote in Leistungspunkten (0-15) auf drei Stellen nach dem Komma und
- die Note als Text gemäß Prüfungsordnung (z.B. *befriedigend*, *gut*, *sehr gut* etc.)

ausgegeben werden.

Schritt 6: Daten in Datei speichern

Speichern Sie die Noten in einer CSV-Datei ab, in der gleichen strukturierten Form wie in der bereitgestellten Datei.

Schritt 7: Lauffähiges Gesamtprogramm

Setzen Sie die einzelnen Aufgabenteile zu einem lauffähigen Gesamtprogramm zusammen und ermöglichen Sie einem User über ein Menü eine einfache Benutzerführung zum Aufruf der unterschiedlichen Funktionen (z.B. e – Eingabe, a – Ausgabe, s – Speichern, ..).

Schritt 8: Programmierstil

Achten Sie auf effiziente Programmierung und einen guten Programmierstil, insbes.

- Verwenden Sie für alle Bezeichner sinnvolle & aussagekräftige Namen
- Benutzen Sie das sog. lowerCamelCase für die Namen ihrer Variablen & Funktionen
- Achten Sie auf korrektes Einrücken
- Verwenden Sie keine globalen Variablen
- Halten Sie die übliche Reihenfolge ein: Makros, Include-Befehle, Funktionsdeklarationen, Hauptprogramm, Funktionsdefinitionen
- Fügen Sie zur besseren Gliederung Leerzeilen und ggf. Linien ein
- Kommentieren Sie ihren Code

Fangen Sie mögliche Fehler im Programm ab und entfernen Sie alle Fehler und Warnungen des Compilers.