Systemprogrammierung - AIN/2

Sommersemester 2022

Übungsaufgabe 4: C Aufzählungen, Strukturen, Varianten, Übersetzungseinheiten

Abgabe bis 19./20.5.2022

Vorbereitung

Speichern Sie die Dateien ⤵ **Makefile**, ⤵ **notenspiegel.c**, ⤵ **notenspiegel-in.txt** und ⤵ **notenspiegel-out.txt** in Ihr Arbeitsverzeichnis von Aufgabe 4.

Programmierung

Das vorgegebene Programm `notenspiegel.c` erstellt einen Notenspiegel. Ergänzen Sie die fehlenden Übersetzungseinheiten wie folgt:

Erstellen Sie eine Übersetzungseinheit `spo3_ain2` mit der folgenden Schnittstelle:

- einer Funktion `bool ist_spo_note(int)`, die `true` liefert, wenn die als Argument übergebene Note laut SPO der HTWG zulässig ist, sonst `false` (zulässig sind die Noten 10, 13, 17, 20, 23, 27, 30, 33, 37, 40, 50)
- einer Funktion `bool ist_ain2_modul(const char *)`, die `true` liefert, wenn es das als Argument übergebene Modul laut SPO für AIN/2 gibt, sonst `false`
- zwei symbolischen Namen `SPO_BESTE_NOTE` und `SPO_SCHLECHTESTE_NOTE` für die beste bzw. schlechteste Note
- einem symbolischen Namen `AIN2_ANZAHL_LEISTUNGEN` für die Anzahl der laut SPO in AIN/2 zu erbringenden Leistungen (Scheine und benotete Prüfungen)

Erstellen Sie eine Übersetzungseinheit `leistung` mit der folgenden Schnittstelle:

- einem Typ `struct leistung` mit eingebetteter anonymer `union` und zugeordnetem `enum`-Typ `leistungsart` (siehe Vorlage in den Vorlesungsunterlagen).

  Die Struktur soll ein Array von Zeichen für den Modulnamen enthalten. Sie dürfen davon ausgehen, dass Modulnamen aus maximal 20 Zeichen bestehen. Definieren Sie die Array-Größe als symbolische Konstante. Mit der zugeordneten `enum` unterscheiden Sie zwischen den Leistungsarten `benotet` und `unbenotet`. Mit der eingebetteten `union` speichern Sie bei benoteten Leistungen eine ganzzahlige Note, bei unbenoteten Leistungen eine Beurteilung in Form eines Einzelzeichens `'B'` für bestanden bzw. `'N'` für nicht bestanden,.

  Deklarieren Sie einen Aliasnamen `leistung` für Ihren `struct`-Typ.

  Wieviel Byte Speicher benötigt eine Variabel vom Typ `leistung`?

- einer Funktion `leistung_einlesen`.

Die Funktion soll von der Standardeingabe eine Leistung einlesen und in die per Ausgabeparameter übergebene Struktur speichern (für Beispieleingaben siehe die Datei `notenspiegel-in.txt`). Beim Einlesen mit `scanf` können Sie zwischen benoteten und unbenoteten Fächern unterscheiden, indem Sie erst mit `%d` eine ganze Zahl zu lesen versuchen, und wenn dies nicht gelingt, mit `%c` ein Einzelzeichen lesen. Die Funktion `scanf` liefert als Rückgabewert, wie viele Werte erfolgreich gelesen werden konnten *(siehe auch `man 3 scanf`)*.

Im Modulnamen sollen nach dem Einlesen alle Unterstriche _ durch Leerzeichen ersetzt werden. Die Unterstriche sollen dazu in einer Schleife mit der Bibliotheksfunktion `strchr` gesucht werden *(siehe auch `man 3 strchr`)*.

Der Rückgabewert von `leistung_einlesen` soll `false` sein, wenn das Lesen nicht gelungen ist, sonst `true`.

- einer Funktion `ausgeben`.

  Die Funktion soll die per Eingabeparameter übergebene Leistung im folgenden Format auf der Standardausgabe ausgeben: in der ersten Spalte steht linksbündig mit Feldbreite 20 der Modulname, bei ungültigem Modulenamen mit dem Zusatz `Fehler: `. Bei einem gültigem Modulenamen folgt dann nach einem Tabulatorzeichen ein L bei benoteten bzw. ein S bei unbenoteten Leistungen und nach einem weiteren Tabulatorzeichen schließlich die Beurteilung (siehe zum Ausgabeformat auch die Datei `notenspiegel-out.txt`).

  Verwenden Sie für die linksbündige Ausgabe mit fester Spaltenbreite des Modulnamens das Format `"%-*s"`. Argumente für dieses Format sind die Spaltenbreite als ganze Zahl, gefolgt vom String *(siehe auch `man 3 printf`).*

  Realisieren Sie die Ausgabe der Note einer benoteten Leistung mit einer privaten Hilfsfunktion `static void ausgeben_benotet(int note)`. Beurteilungen zwischen 10 und 40 sowie die 50 werden als 1,0 bis 4,0 und 5,0 ausgegeben. Sonstige ganze Zahlen werden mit dem Zusatz `Fehler: ` so ausgegeben, wie sie eingegeben wurden.

  Realisieren Sie die Ausgabe der Beurteilung einer unbenoteten Leistung mit einer privaten Hilfsfunktion `static void ausgeben_unbenotet(char status)`. Die Beurteilung `'B'` wird als `bestanden` und die Beurteilung `'N'` als `nicht bestanden` ausgegeben. Sonstige Beurteilungen werden mit dem Zusatz `Fehler: ` als Einzelzeichen ausgegeben.

Test und Qualitätssicherung

Verwenden Sie zum Testen die folgenden Befehle und probieren Sie verschiedene Eingaben, insbesondere auch `hallo` als einzige Eingabe:

```
make notenspiegel
make cppcheck
./notenspiegel
valgrind ./notenspiegel
```

Führen Sie auch die folgenden automatisierten Tests aus:

```
valgrind ./notenspiegel < /dev/null
valgrind ./notenspiegel < notenspiegel-in.txt
./notenspiegel < notenspiegel-in.txt > out.txt
diff -Z notenspiegel-out.txt out.txt
```

- `valgrind` darf keine Fehler und `diff` keine Unterschiede melden.
- `cppcheck` sollte keine Probleme melden.

Bessern Sie gegebenenfalls nach.

Abgabe

Führen Sie Ihr Programm mit den automatisierten Tests vor.

Hinweis:
Der Compiler `gcc` darf für Ihr Programm keine Fehler oder Warnungen mehr ausgeben.
Ihr Programm muss außerdem ordentlich formatierte sein. Bessern Sie die Formatierung gegebenenfalls mit `astyle` nach:
`astyle -p -H --style=ansi *.[ch]`

Freiwillige Zusatzaufgabe *(1 Bonuspunkt)*

cppcheck verlangt, dass Sie bei `scanf` mit dem Format `%s` die maximale Anzahl zu lesender Zeichen festlegen, in `leistung.c` also sinnvollerweise `%20s`. Die `20` kommt dann zweimal im Programm vor: in der Deklaration der symbolischen Konstanten in `leistung.h` und in der Formatspezifikation in `leistung.c`. Bei Programmänderungen wird leicht eine der beiden Stellen übersehen, was dann zu Laufzeitfehlern führt.

Das Problem lässt sich mit der sogenannten Stringification des C-Präprozessors lösen. Ersetzen Sie in der Formatspezifikation von `scanf` die `20` durch den stringifizierten Wert Ihrer symbolischen Konstanten. Testen Sie das geänderte Programm per `valgrind` mit zu langen Modulnamen.

Hinweis: Googlen Sie "C macro stringification".

Prof. Dr. H. Drachenfels
Hochschule Konstanz** - **Impressum** - **Datenschutzerklärung

Letzte Änderung: 23.2.2022