Übung Nr. 2:

Es soll eine einfache Liste von Fächern, Professoren und Studierenden eines fiktiven Informatik-Studiengangs erstellt werden. Der größte Teil des benötigten Codes und das dazugehörige MS-VC-Projekt sowie ein ausführbares Exemplar des Ergebnis-Programms wird unter http://homepages.thm.de/christ/ bereitgestellt, die noch fehlenden Teile (insg. ca. 15 Zeilen) sind an markierte Stellen im gelieferten Code zu setzen und werden im folgenden in vier Arbeitsabschnitten beschrieben. Es wird empfohlen, diese Abschnitte in der u.a. Reihenfolge zu behandeln.

1. Extraktion des Programm-Namens aus dem Datei-Pfad

Das Programm soll sich beim Start mit seinem Namen ohne Pfad melden (Abb. 1). Hierzu soll der Pfad zur ausführbaren Datei aus dem ersten Argument von `main()` extrahiert werden. Die dazu vorgesehene Funktion ist bereits im Einsatz, sie hat aber keinen Effekt, weil der benötigte Code (markiert mit `MORE_STRUCT1`) fehlt.

.

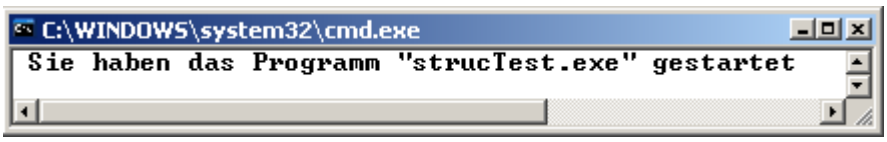

Abb. 1 Startfenster

2. Erzeugung von Pseudo-Zufallszahlen in vorgegebenem Zahlenintervall

Zur Erzeugung zufälliger (engl.: random) Namen und Noten wird die Funktion `rand()` aus der C-Standardbibliothek `stdlib.h` verwendet; sie bildet nach einer festen Rechenvorschrift ganze Zahlen zwischen 0 und `RAND_MAX` (=32767), an denen kein Bildungsgesetz zu erkennen ist. Will man vermeiden, daß bei jedem Programmlauf dieselbe Zahlenfolge entsteht, so kann man zur Initialisierung der Zahlenfolge die Funktion `void srand(unsigned int)` einsetzen. Als Argument wird meist die aktuelle Zeit eingesetzt, so daß zum Pseudo-Zufall der zufällige Zeitpunkt hinzukommt. Diese Zusammenhänge sind bereits im gelieferten Code eingearbeitet. (Sie sollten trotzdem beachtet werden...)

Wird ein begrenzter Zahlenbereich (engl.: range) als Ergebnis von `rand()` gewünscht, so findet sich z.B. in der Hilfe des MSDN (Microsoft Developer Network) der Vorschlag, der hier als Funktion `int randRang(int rangeMin, int rangeMax)` implementiert wurde. Dieser ist jedoch offenbar nur geeignet, wenn die untere Grenze des Zahlenintervalls null ist; das soll korrigiert werden.

Wird der vorhandene Code mit definierter Präprozessor-Konstanten `DEBUG` compiliert, so erscheint im Ausführungsfenster als erstes eine Reihe von Angaben über die Größe der hier verwendeten Strukturen; danach wird bei jedem Tastendruck eine neue Pseudo-Zufallszahl ausgegeben, wobei die o.a. Initialisierung umgangen wird; d.h., es entsteht bei jedem Start dieselbe Zahlenfolge (Abb. 2). Zum Abbruch und zur Fortsetzung des Programmlaufs muß die Taste <ESC> gedrückt werden.

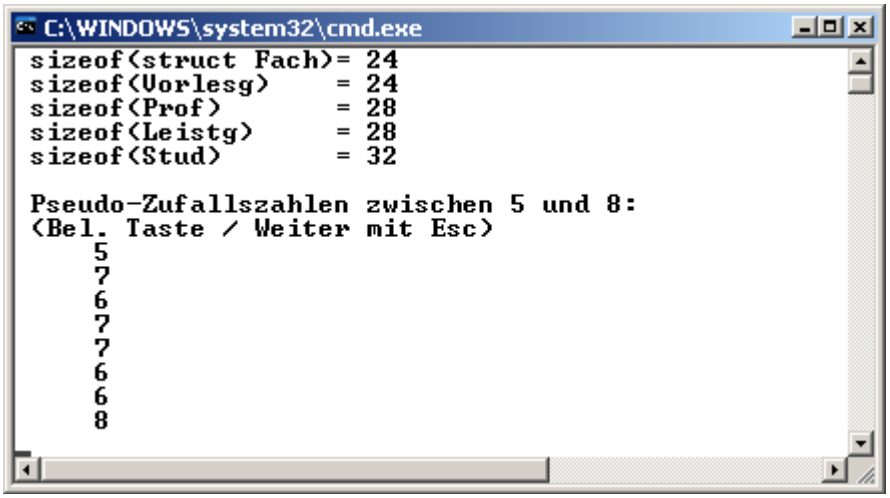

Abb. 2 Zufallszahlen

Mit Hilfe dieser Funktionalität soll der Code von `randRang()` angepaßt werden (Markierung `MORE_STRUCT2`).

3. Bildung zufälliger Studentendaten

Die Funktion erhält die Adresse des Datensatzes eines fiktiven Studenten. Als Name werden Initialen eingesetzt; sie bestehen aus einem zufälligen Vornamen-Initial und einem in alphabetischer Reihenfolge genommenen Anfangsbuchstaben für einen Familiennamen. Zur Vermeidung eines Überlaufs wird das Alphabet zyklisch verwendet. (Man sollte versuchen, die Arbeitsweise der hier mehrmals eingesetzten Funktion `CYCLIC(Nr,A,Z)` zu verstehen.)

Während allen Studierenden unterstellt wird, sie hätten drei Leistungen zu absolvieren, wird jedem eine individuelle Matrikelnummer zugewiesen. (Wie wird letzteres bewerkstelligt?)

Zur Vervollständigung der Funktion (Markierung `MORE_STRUCT3`) fehlt noch die Speicherplatz-Reservierung und die Zuweisung der Leistungsdaten. Die drei benötigten Leistungen sollen den Studierenden aus der Reihe der gespeicherten Vorlesungen so zugeordnet werden, daß Matr. 1 die Vorlesungen mit den Indizes 0-2 hat, Matr. 2 jene mit Indizes 1-3 etc.. Bei Erreichen der letzten Vorlesung wird die Zuordnung zyklisch wiederholt.

Als Note wird schließlich für jedes Fach eine zwischen 2 und 3 zufällig gewählte Note mit einem ebenfalls zufällig gebildeten Versatz, der 0 oder ±0,3 betragen kann.

4. Plausibilitätskontrolle beim reservierten Speicherplatz

Als beispielhafte Plausibilitätskontrolle soll (Markierung `MORE_STRUCT4`) überprüft werden, ob der Speicherplatz, der für die erste Vorlesung des ersten Professors reserviert wurde, exakt die geforderte Größe hat. Der vorläufig verwendete Code ist an dieser Stelle offenbar nicht hilfreich.

Nach erfolgreicher Behandlung der o.a. Aufgaben sollte das Programm wie das bereitgestellte Funktionsmuster arbeiten.