Frankfurt University of Applied Sciences
Fachbereich 2: Informatik und Ingenieurwissenschaften

Einführung in die Programmierung mit der Programmiersprache C (**EifP**, WS2025/26)

Prof. Dr. Thomas Gabel
Roman Ahmad, Emre Özöner, Gina Romanazzi

Aufgabenblatt 8

Aufgabe 35: Fakultät

Inhalte und Sprachkonzepte der vorliegenden Aufgabe

- Verwandtschaft der `for`-, `while`-Schleife

Aufgabenstellung

a) Schreiben Sie ein Programm, das für die Zahlen von 0 bis 50 die Fakultät berechnet. Nutzen Sie hierzu eine Zählschleife.

b) Geben Sie eine Einschätzung ab, bis zu welcher Zahl (aus $[0, 50]$ die von Ihnen berechnete und angezeigte Fakultät korrekt ist. Warum sind ab einem gewissen Punkt die Fakultätsberechnungen Ihres Programmes fehlerhaft?
Hinweis: Denken Sie über die Größe der berechneten Fakultätswerte nach und ziehen Sie die Nutzung alternativer Datentyen für die Abspeicherung Ihres Rechenergebnisses in Erwägung.

c) Schreiben Sie das Programm aus Aufgabenteil a) so um, dass eine abweisende Schleife zum Einsatz kommt.

Aufgabe 36: Familie Meier

Inhalte und Sprachkonzepte der vorliegenden Aufgabe

- praktische Anwendung logischer Operatoren und boole'scher Ausdrücke

Aufgabenstellung

Familie Meier ist zu einer Geburtstagsfeier eingeladen. Leider können sich die Familienmitglieder (Alfons, Beate, Christian und Doris) nicht einigen, wer hingeht und wer nicht. In einer gemeinsamen Diskussion kann man sich jedoch auf die folgenden Grundsätze verständigen:

1. Mindestens ein Familienmitglied geht zu der Feier.
2. Alfons geht auf keinen Fall zusammen mit Doris.
3. Wenn Beate geht, dann geht Christian mit.
4. Wenn Alfons und Christian gehen, dann bleibt Beate zu Hause.
5. Wenn Alfons zu Hause bleibt, dann geht entweder Doris oder Christian.

Helfen Sie Familie Meier, indem Sie ein Programm `familie_meier.c` erstellen, das alle Gruppierungen ermittelt, in denen Familie Meier zur Feier gehen könnte.
Hinweis: Erstellen Sie alle denkbaren Kombinationen und prüfen Sie jede darauf, ob sie alle der oben angegebenen Grundsätze erfüllt.

Aufgabe 37: Grundlagen von Feldern

Inhalte und Sprachkonzepte der vorliegenden Aufgabe

- Initialisieren von Feldern
- Feldinhalt bearbeiten
- globale Variablen
- Funktionen

Aufgabenstellung

Schreiben Sie ein C-Programm zum Umgang mit Feldern und erweitern Sie es gemäß der folgenden Teilaufgaben.

a) Legen Sie ein Feld von von Integer-Zahlen an. Das Feld soll die Größe 10 haben und als globale Variable definiert sein.

b) Lassen Sie in Ihrem Hauptprogramm die Werte für die 10 Elemente des Feldes vom Benutzer eingeben.

c) Schreiben Sie nun eine Funktion `void ausgabe()`, mit der Sie alle Elemente des Feldes ausgeben. Rufen Sie diese aus Ihrem Hauptprogramm heraus auf.

d) Schreiben Sie nun eine C-Funktion `int summe()`, die die Summe der Elememente in Ihrem Feld berechnet und zurückgibt. Lassen Sie sich in Ihrem Hauptprogramm die Summe der Elemente in Ihrem Feld ausgeben.

e) Erweitern Sie Ihr Hauptprogramm nun um die Funktionalität, dass sämtliche Elemente Ihres Feldes invertiert werden. Berechnen Sie (nach erfolgter Invertierung) die Summe der Elemente Ihres Feldes und geben Sie diese aus.

Aufgabe 38: Alphabetisches

Inhalte und Sprachkonzepte der vorliegenden Aufgabe

- Felder
- Operationen auf Feldelementen

Aufgabenstellung

Zu entwickeln ist ein Programm `isalphabetic.c`, das:

a) prüft, ob ein Feld nur alphabetische Elemente enthält und eine entsprechende Meldung anzeigt,

b) nicht alphabetische Elemente nicht nur entdeckt, sondern auch durch ein '*'-Zeichen ersetzt und das neue Feld ausgibt.

Testen Sie Ihr Programm mit den folgenden schon initialisierten Arrays fester Größe (10):

- `char word1[] = {'a', 'b', 's', 'D', 'F', 'G', '1', 'j', 'k', '2', '\0'};`
- `char word2[] = {65, 69, 90, 83, 102, 78, 71, 98, 119, 48, 0};`
- `char word3[] = "lllllllll";`

Aufgabe 39: Zufallsfarben

Inhalte und Sprachkonzepte der vorliegenden Aufgabe

- Nutzung von Bibliotheken, Anwendung mehrerer Kompilationseinheiten (vgl. Tutorial vom vorangegangenen Aufgabenblatt)
- Zufallsgenerator

Aufgabenstellung

In dieser Aufgabe ist ein Programm zu entwickeln, das den Inhalt der Konsole (Terminal, xterm) auf farbenfrohe Weise "zersetzt". Hierfür ist die Ihnen im campUAS-Kurs unter `U8_Resources.tar.gz` zur Verfügung gestellte Kompilationseinheit "Advanced Console Output" (kurz: ACO) in Form der beiden Dateien `advanced_console_output.[ch]`[1] zu verwenden.

[1] Mit der Schreibweise `dateiname.[ch]` wird Bezug genommen auf sowohl die c-Datei `dateiname.c` als auch auf die zugehörige Header-Datei `dateiname.h`. Dies ist eine gängige, verkürzende Schreibweise.

a) Machen Sie sich mit der erwähnten Kompilationseinheit "Advanced Console Output" (ACO) vertraut, mit der Sie u.a. die Ausgabe im Terminal farbig gestalten aber auch Texte fett, kursiv oder unterstrichen ausgeben können. Gehen Sie dabei wie folgt vor:

   - Laden Sie das erwähnte Archiv `U8_Resources.tar.gz` herunter.
   - Entpacken Sie das Archiv und kopieren Sie die enthaltenen Dateien in ein Verzeichnis, in dem Sie die diese Aufgabe lösen.
     Hinweis: Ein `tar.gz`-Archiv können Sie mit dem Befehl `tar` wie folgt entpacken: `tar xvfz namederdatei.tar.gz`
   - Erkunden Sie die Inhalte des entpackten Archivs, indem Sie sich mit dem enthaltenen Quellkode vertraut machen; lesen Sie sich grob die Inhalte und Kommentare in den Quellkodedateien durch.

b) Erstellen Sie das im Archiv enthaltene Testprogramm[2] `aco_test`, welches Ihnen einen groben Eindruck von den Möglichkeiten der ACO-Bibliothek vermittelt:

   - Kompilieren Sie die Kompilationseinheit `advanced_console_output.c` und erstellen Sie so das Modul `advanced_console_output.o`.
   - Kompilieren Sie das Testprogramm `aco_test.c` und erstellen Sie so das Modul `aco_test.o` (jenes enthält auch das Hauptprogramm `main()`).
   - Linken Sie die beiden entstandenen Object-Dateien zu einem ausführbaren Programm zusammen. Lassen Sie das erzeugte Programm laufen!

c) Schreiben Sie nun ein eigenes Programm namens `aco_zufallsfarben.c`, das die ACO-Bibliothek verwendet und das folgende Funktionalität realisiert:

   - Der Zufallsgenerator wird systemzeitbasiert initialisiert.
   - Die Größe der Konsole (Breite und Höhe wird ermittelt).
   - Eine abweisende Endlosschleife wird betreten, innerhalb derer
     - der Cursor an eine zufällige Position in der Konsole gesetzt wird (→ hierzu ist eine passende Funktion aus der Kompilationseinheit "Advanced Console Output" zu verwenden),
     - die Hintergrundfarbe der Ausgabe zufällig gesetzt wird (→ hierzu ist eine passende Funktion aus der Kompilationseinheit "Advanced Console Output" zu verwenden),

[2] In dieser Teilaufgabe wenden Sie unmittelbar Ihre Kenntnisse aus dem Tutorial vom letzten Aufgabenblatt an.

– ein Leerzeichen ausgegeben wird,
– der Ausgabepuffer ausgegeben wird (“geflusht” via `fflush()`),
– das Programm eine Millisekunde pausiert (via `usleep()`).

Hinweis: Ihr Hauptprogramm (Teilaufgabe c)) wird kaum mehr als 10 Zeilen Quellkode umfassen. Ein Beispiel für ein ausführbares Programm (für Teilaufgabe c)) ist im erwähnten Archiv unter dem Namen `aco_zufallsfarben` ebenfalls enthalten.