IFM 3.3 Systemprogrammierung, W19

Blatt 01: Einrichtung Raspi, Einstieg C [ANSI-C] (10 Punkte)

Praktikum: 17.10.2019

Carsten Gips, Birgit C. George, FH Bielefeld

---

1 Vorbereitung: Einrichtung des Raspberry Pi

Auf der SD-Karte Ihres Raspi-Leihsets ist bereits Raspbian vorinstalliert. Richten Sie mit Hilfe des Handouts zu VL01 sowie der offiziellen Raspberry Pi Documentation Ihr Raspi-Set ein:

1. Passen Sie die Sprache, d.h. Lokalisation und Tastaturlayout an Ihre Bedürfnisse an
2. Aktivieren Sie die SPI-Schnittstelle
3. Vergeben Sie einen Hostnamen nach dem Muster `swlab-raspi-INITIALENDERGRUPPENMITGLIEDER`
4. Ändern Sie das Passwort
5. Aktualisieren Sie das Tool `raspi-config` sowie das vorinstallierte Raspbian-OS

   Hinweis: Dies geht i.d.R. sehr schnell, kann aber u.U. bis zu zwei Stunden dauern!
6. Installieren Sie über die Paketverwaltung folgende Softwarepakete nach: `build-essential cmake googletest valgrind wiringpi doxygen-gui libsqlite3-dev libxml2-dev libxml++2.6-dev libncurses5-dev`
7. Vervollständigen Sie die Installation der Google-Test- und Google-Mock-Bibliotheken:
   1. `cd /usr/src/googletest/`
   2. `sudo cmake CMakeLists.txt`
   3. `sudo make install`
8. Installieren Sie cURL mit dem klassischen **Unix-Dreisprung**: `./configure`, `make`, `make install`

   Laden Sie von curl.haxx.se/download ein Archiv der aktuellen Sourcen von cURL herunter. Entpacken Sie das Archiv, wechseln Sie in den Ordner und führen Sie den Unix-Dreisprung aus: `./configure && make && sudo make install` ...

2 Aufgaben

2.1 Limits kennen: Datentypen, Wertebereiche (1 Punkt)

Schreiben Sie ein C-Programm, welches die größtmögliche `unsigned int` Zahl auf Ihrem System berechnet und ausgibt.

Verwenden Sie hierzu **nicht** die Kenntnis der systemintern verwendeten Bytes (`sizeof`, ...). Nutzen Sie auch nicht die Konstanten/Makros/Funktionen aus `limits.h` oder `float.h` oder anderen Headerdateien!

Thema: Einstieg in C-Programmierung: Grenzen der Datenbereiche

2.2 Sinus: Schleifen, Umgang mit Funktionen aus der Standard-Bibliothek (1 Punkt)

Schreiben Sie ein C-Programm, welches für alle Winkelwerte zwischen 0 und 360 Grad in 10-Grad-Schritten den Sinus berechnet und auf drei Stellen nach dem Komma genau ausgibt. Geben Sie jeweils auch den Winkel mit aus, beispielweise

```
Winkel:  0 Grad => Sinus-Funktionswert: 0.000
Winkel: 10 Grad => Sinus-Funktionswert: 0.174
...
Winkel: 90 Grad => Sinus-Funktionswert: 1.000
...
```

Schreiben Sie zwei Varianten des Programms: Nutzen Sie einmal eine `for`- und einmal eine `while`-Schleife.

Hinweis: Wo finden Sie die Sinus-Funktion? Wie teilen Sie das dem Compiler (und Eclipse/Ihrer IDE) mit?

Hinweis: Welchen Datentyp nutzen Sie sinnvollerweise für die Iteration? Welchen Datentyp erhält die Sinus-Funktion als Eingabe? Nutzt die Sinus-Funktion als Eingabe Grad oder Bogenmaß (Radian)?

Thema: Einstieg in C-Programmierung: Schleifen in C; Nutzung von man-Pages: `man 3 sin`

2.3 Ein- und Ausgabe, Schleifen (1 Punkt)

Schreiben Sie ein C-Programm, welches für die Eingabe einer positiven ganzen Zahl n folgende Ausgaben produziert. Nutzen Sie dazu die Funktionen `scanf()` und `printf()` aus `stdio.h`. Nutzen Sie unterschiedliche Schleifenkonstrukte!

1. Ausgabe:

   ```
   ***...***
   ```

   (n Sterne in einer Reihe)

2. Ausgabe:

   ```
   *
   **
   ***
   ...
   ***...***
   ```

   (linksbündige Pyramide, beginnend mit einem Stern, je Zeile ein Stern mehr, in Zeile n dann Reihe mit n Sternen)

3. Ausgabe:

   ```
      *
     ***
     ...
   ***...***
   ```

   (zentrierte Pyramide, beginnend mit einem Stern, in Zeile i entsprechend $2i - 1$ Sterne, letzte Reihe mit n Sternen; n muss ungerade sein!)

Thema: Einstieg in C-Programmierung: Schleifen, Ein- und Ausgabe

2.4 Bit-Operationen (1 Punkt)

Schreiben Sie eine C-Funktion, welche eine Zahl zwischen 0 und 255 übergeben bekommt und prüft, ob in dieser Zahl ein bestimmtes Bit gesetzt ist. Die Nummer des Bits wird dabei ebenfalls als Funktionsparameter übergeben. Anschließend soll das Bit gesetzt werden (Wert 1) und danach gelöscht (Wert 0) werden. Das Ergebnis der ursprünglichen Prüfung soll als Wahrheitswert zurückgeliefert werden.

Geben Sie nach jeder Operation/Prüfung das Ergebnis aus.

Hinweis: Das zu betrachtende Bit und die Zahl können im Hauptprogramm fest codiert werden. Sie können natürlich auch beide Zahlen im Hauptprogramm als Tastatureingabe einlesen.

Thema: Einstieg in C-Programmierung: Bit-Operationen

2.5 Casts: Implizite und explizite Typumwandlungen (1 Punkt)

Geben Sie alle unnötigen bzw. unsinnigen Typumwandlungen (casts) in folgendem Codefragment an.

```
double x;
x = (float)7/4;                 x = (double)(7/4);
x = (double)(7/(float)4);       x = (double)7/4;
x = (double)(7/4.0);            x = (double)7.0f/4;
```

2.6 Personalverwaltung (5 Punkte)

Schreiben Sie ein Programm in **ANSI-C** für eine einfache Personalverwaltung.

a) Erstellen Sie einen Aufzählungstyp `division`, der die Unternehmensbereiche `MANAGEMENT`, `DEVELOPMENT`, `PRODUCTION`, `MARKETING` und `SALES` umfasst.

b) Definieren Sie einen neuen Datentypen `employee`, der folgende Informationen enthält:

- `last_name` (Char-Array fester Länge)
- `first_name` (Char-Array fester Länge)
- `id` (positive ganze Zahl)
- `department` (vom Enum-Typ `division`)
- `salary` (reelle Zahl, einfache Genauigkeit)

c) Definieren Sie ein *globales* Feld `company`, welches 10 Angestellte umfassen kann und welches Sie mit mindestens 4 verschiedenen Angestellten initialisieren.

Definieren (und nutzen) Sie für die Länge des Arrays die (Präprozessor-) Konstante `STAFFCOUNT`.

Hinweis: Das Feld `company` ist eine globale Variable. Wird diese automatisch initialisiert? Welche Werte stehen dann da drin? Wie erkennt man fehlende Angestellte, also leere Array-Plätze?

d) Definieren Sie eine *parameterlose* Funktion `calculate_average_salary()`, die zu dem globalen Feld `company` aus dem vorigen Aufgabenteil das Durchschnittsgehalt über alle Angestellten berechnet und als Ergebnis zurück liefert.

e) Definieren Sie eine weitere *parameterlose* Funktion `change_salary()`, die für alle Angestellten im globalen Feld `company` eine Gehaltserhöhung durchführt. Das Management soll 30%, die Entwickler 10%, Produktion und Vertrieb sollen jeweils 1% und das Marketing 2% Gehaltserhöhung erhalten. Es kann auch negative Gehaltserhöhungen geben (beispielsweise -10%). In diesem Fall soll sich das Gehalt entsprechend verringern.

Verwenden Sie für diese Funktion keine `if`- oder `switch`-Anweisung! Überlegen Sie, was Sie mit dem im ersten Teil definierten Aufzählungstyp anstellen können :-)

f) Definieren Sie eine Funktion `print_employee()`, mit der Sie **einen** (per Funktionsparameter übergebenen) Angestellten vernünftig formatiert auf der Konsole ausgeben können. Nutzen Sie dazu die Funktion `printf()` aus `stdio.h`.

Schreiben Sie eine *parameterlose* Funktion `print_personnel()`, die das globale Feld `company` unter Zuhilfenahme der Funktion `print_employee()` ausgibt.

Hinweis: Die neu definierten Typen sollen sich wie eingebaute Typen verwenden lassen, beispielsweise bei der Deklaration von Variablen: `division x;` bzw. `employee y;` ...

Hinweis: Verwenden Sie in dieser Aufgabe noch **keine** Pointer bzw. dynamische Speicherverwaltung! Der Stoff von VL01 und VL02 reicht zum Bearbeiten der Aufgabe.

Thema: Umgang mit Basisdatentypen und Strukturen und Funktionen