Universität Potsdam
Institut für Informatik

Sommersemester 2014

Praxis der Programmierung

Aufgabenblatt Woche 2

1. Kopieren Sie die Datei `/home/rlehre/Woche_02/formatelemente.c` und
   a) ergänzen Sie den Quellcode zu einem fertigen C-Programm, so dass der Compiler keine Warnungen ausgibt,
   b) führen Sie das Programm aus und analysieren Sie den Quellcode,
   c) verbessern Sie die Ausgabe, so dass der Eingabeprompt der Shell nach Ausführung des Programms in einer eigenen Zeile ausgegeben wird.

   Notieren Sie sich die Bedeutung der Formatelemente.

2. Konsultieren Sie die Manual-Seite von `printf` mit dem Ziel, ein C-Programm zu schreiben, das die Zeile

   ```
   Ich kenne die Formatelemente zu 100%.
   ```

   ausgibt.

3. Kopieren Sie die Datei `/home/rlehre/Woche_02/typengroesse.c` und
   a) Analysieren Sie die Syntax und Bedeutung des `sizeof`-Operators!
   b) Ergänzen Sie das Programm, so dass Sie eine Übersicht über die Größe der Speicherbereiche für jeden der elementaren Datentypen erhalten.

4. Schreiben Sie ein C-Programm `for1.c`, dass zuerst die geraden Zahlen von 12 bis 0, dann die ungeraden Zahlen von -1 bis -13 und zum Schluss die ersten 10 Quadratzahlen mit Hilfe von `for`-Schleifen ausgibt:

   ```
   12, 10, 8, 6, 4, 2, 0
   -1, -3, -5, -7, -9, -11, -13
   0, 1, 4, 9, 16, 25, 36, 49, 64, 81
   ```

   Das Ausgabeformat sollte diesem Beispiel folgen.

5. Unter Verwendung der Formel

$$\text{Grad Celsius} = \frac{5}{9} \cdot (\text{Grad Fahrenheit} - 32)$$

soll eine Temperaturtabelle in folgender Form auf `stdout` ausgegeben werden:

```
Fahrenheit  Celsius
  0          -17
 20           -6
 40            4
        ⋮
300          148
```

Verwenden Sie eine `while`-Schleife und `int`-Variablen!

6. Die Standardfunktion `getchar()` aus der Header-Datei `stdio.h` liest *einzelne* Zeichen von `stdin` und gibt die gelesenen Zeichen als `int`-Wert zurück. Dabei wird die Standardeingabe zeilengepuffert, d. h., `getchar()` wartet, bis ein Newline-Zeichen eingegeben wird. Wird ein Dateiende erreicht oder von der Tastatur die Konstante `EOF` (gleichzeitiges Drücken von `<Strg> <d>`) eingegeben, so ist der Rückgabewert `-1`.
   a) Kopieren Sie die Datei `/home/rlehre/Woche_02/char_in.c` in Ihr Arbeitsverzeichnis.
   b) Analysieren Sie den Quellcode und das Verhalten, wenn das Programm ausgeführt wird.
   c) Schreiben Sie ein C-Programm, das mit `getchar()` solange Zeichen einliest, bis *Strg-D* eingegeben wird und am Ende die Anzahl der eingegebenen Zeilen ausgibt.
   Hinweis: Das Enter-Zeichen wird jedesmal mitgezählt.

Funktionen

Funktionen sind ein bequemer Weg, den Quelltext in übersichtliche Teile zu gliedern. Es ist guter Stil größere Programmteile in Funktionen auszulagern. Das hat einen praktischen Grund: Kleine Aufgaben lassen sich einfacher duchdenken und programmieren. Außerdem kann man auf diese Weise einzelne Codeblöcke wiederverwerten, ohne sie erneut kopieren zu müssen - ein einfacher Funktionsaufruf reicht.
Funktionen werden in C durch ihre Struktur definiert. Mit dem Schlüsselwort `return` *können Funktionen Werte zurückliefern.*

```
1 <Rückgabetyp> <Name>([<Parameterliste>]) {
2     <Anweisungsfolge>
3 }
```

Einige Beispiele: [1]

```c
#include <stdio.h>

void hello() {
    printf("Hallo Welt!\n");
}

int main(void) {
    hello();
    return 0;
}
```

```c
#include <stdio.h>

int mult(a, b) {
    return a * b;
}

int main(void) {
    int x = mult(2, 3);
    printf("2*3 = %d", x);
    printf("\n");
    return 0;
}
```

7. Schreiben Sie Funktionen, die die folgenden Aufgaben erfüllen. Testen Sie Ihre Funktionen in einem C-Programm.
   a) Zwei vom Benutzer eingelesene Zahlen sollen der Funktion übergeben, und der Bereich zwischen diesen Zahlen soll in der Funktion auf der Standardausgabe ausgegeben werden.
   b) Eine übergebene Zahl soll quadriert und zurück gegeben werden.
8. Schreiben Sie ein C-Programm, das eine Funktion zum Berechnen des Dreifachen vom Quadrat einer als Parameter übergebeben ganzen Zahl enthält. Diese Funktion wird von der `main`-Funktion aufgerufen, welche das Ergebnis der Berechnung auf der Standardausgabe ausgibt.

[1] mehr Beispiele gibt es auf `https://de.wikibooks.org/wiki/C-Programmierung:_Funktionen`

Schleifenkontrollen

Sie haben bereits gesehen, wie man eine Schleife bei einer bestimmten Eingabe beenden kann. Um die Kontrolle von Schleifen einfacher zu gestalten, gibt es die zwei Kontrollanweisungen `break` *und* `continue`*.*

9. Erschließen Sie sich den Unterschied von `break` und `continue` am nachfolgenden Beispiel:

```c
#include <stdio.h>

int main(void) {
    int n = 0;
    while (n >= 0) {
        printf("=== Schleifenblock beginnt ===\n");
        printf("Ganze Zahl eingeben: n = ");
        scanf("%d", &n);
        if (n == 0) {
            break;
        } else if (n % 2 == 0) {
            continue;
        }
        printf("Zahl ist ungerade\n");
        printf("=== Schleifenblock endet ===\n");
        getchar();
    }
    if (n < 0) {
        printf("Negative Zahl eingegeben.\n");
    }

    printf("Skript wird beendet.");
    printf("\n");

    return 0;
}
```

a) Wohin springt die Ausführung, wenn `continue` aufgerufen wird?

b) Was passiert, wenn die Bedingung der `while`-Schleife nicht mehr erfüllt ist?

c) Welchen Effekt hat `break`?