Christian-Albrechts-Universität zu Kiel
Institut für Informatik

Prof. Dr.-Ing. Norbert Luttenberger, Hagen Peters, Miro Spönemann

Institut für Informatik

1. Übung zur Vorlesung „Organisation und Architektur von Rechnern“

Programmierung in C
Abgabe am Montag, 18. April – 23:59

Die Übungsaufgaben dienen zur Festigung und Selbstüberprüfung der C-Kenntnisse und sind freiwillig zu bearbeiten. Es wird pro erfolgreich gelöster Aufgabe **ein Bonuspunkt** angerechnet. Die Lösungen können nach erfolgter Anmeldung zu einer Übungsgruppe im iLearn abgegeben werden (ab April möglich) oder während der betreuten Rechnerzeit in der ersten Vorlesungswoche im Grundausbildungspool (HRS3 R.105, Do 14.4., 14–16 Uhr) einem Betreuer präsentiert werden. Die betreute Rechnerzeit eignet sich außerdem sehr gut um Fragen zu stellen und die Aufgaben mit Unterstützung zu bearbeiten.

Abgabe per iLearn. Für jeden Aufgabenteil soll eine Datei hochgeladen werden. Falls nicht anders angegeben, soll jede Datei ein vollständiges C Programm enthalten, das fehlerfrei kompiliert und ausgeführt werden kann. Jedes C Programm soll ausführlich kommentiert werden, um die Lösung zu erläutern und die in den Aufgaben vorkommenden Fragen zu beantworten. Sie dürfen die Aufgaben auch in Kleingruppen lösen, aber jeder muss selbst die Lösungen hochladen.

Abgabe zur betreuten Rechnerzeit. Die Lösung jeder Aufgabe kann einzeln einem Betreuer präsentiert werden, der dies notiert und später für Sie anrechnet. Sie dürfen die Aufgaben auch in Zweiergruppen lösen.

Aufgabe 1 - Compiler

1 Punkt

(a) Schreiben Sie ein C-Programm, das den Text “*Hello, World!*” am Bildschirm ausgibt und speichern Sie es in einer Datei `hello.c`. Kompilieren Sie das Programm mit dem `gcc` Compiler und führen Sie es aus.

Erzeugen Sie durch spezielle Optionen des `gcc` alle Zwischenergebnisse des Kompilierlaufs:

(b) `hello.i`: Ergebnis nach Präprozessor **(c)** `hello.s`: Ergebnis nach Compiler (Assembler Code) **(d)** `hello.o`: Ergebnis nach Assembler (binäre Objekt-Datei) **(e)** `hello`: Ergebnis nach Linker (ausführbares Programm)

Aufgabe 2 - Datentypen und Funktionen

1 Punkt

Schreiben Sie eine Funktion namens `c2f`, die Grad Celsius in Grad Fahrenheit umrechnet, und eine zweite namens `f2c`, die das Umgekehrte tut. Dabei soll Celsius immer mit ganzzahligem Typ (`int`) und Fahrenheit immer mit Fließkomma-Typ (`double`) repräsentiert werden. Die mathematische Formel lautet $T_F = T_C \cdot \frac{9}{5} + 32$. *Hinweis:* Achten Sie auf das unterschiedliche Verhalten der Division je nach Datentypen.

Fügen Sie als Test für Ihre Funktionen eine `main`-Prozedur mit folgendem Inhalt hinzu:

```
printf("%d\n", f2c(c2f(5)));
```

Wenn Ihre Funktionen korrekt sind, muss das Programm eine 5 ausgeben.

Aufgabe 3 - Ein- und Ausgabe

1 Punkt

Schreiben Sie eine Funktion `printID(int i, double d)`, die eine Kombination aus einem `int` und einem `double` Wert auf der Console ausgibt. Geben Sie die Werte in einer Zeile jeweils rechtsbündig immer mit Vorzeichen (auch für positive Zahlen) und Fließkommazahlen nur auf 2 Kommastellen genau aus.

Ergänzen Sie als Test eine `main`-Prozedur, die eine Zeichenfolge aus der Tastatur einliest, diese in einen `int` Wert umwandelt, daraus mit der Funktion `c2f` einen entsprechenden `double` Wert berechnet und mit der Funktion `printID` beide Werte ausgibt. *Hinweis:* Verwenden Sie `gets` zum Einlesen und `atoi` zum Umwandeln.

Aufgabe 4 - Schleifen

1 Punkt

(a) Schreiben Sie ein Programm, das mittels einer `for` Schleife alle ganzzahligen Temperaturwerte von $-10°$ bis $+30°$ Celsius mit `c2f` nach Fahrenheit umrechnet und mit `printID` jedes Wertepaar ausgibt.

(b) Schreiben Sie ein Programm, das mittels einer `while` Schleife von $10°$ Fahrenheit beginnend in $2,5°$ Schritten ansteigend Werte mit `f2c` nach Celsius umrechnet und mit `printID` jedes Wertepaar ausgibt. Die Schleife soll beendet werden, wenn der umgerechnete Wert 30°C übersteigt.

Aufgabe 5 - Verzweigung

1 Punkt

Verändern Sie das Programm aus Aufgabe 4(b), indem Sie die Funktion `printID` so erweitern, dass die Ausgabe der Zahlenwerte nur dann durchgeführt wird, wenn der ganzzahlige Celsius Wert im Intervall $[10, 20]$ liegt. Andernfalls soll der Text "*Value out of range.*" ausgegeben werden.

Aufgabe 6 - Arrays

1 Punkt

Schreiben Sie eine Funktion `printA(double a[], int n)`, welche `n` Werte aus dem Array `a` nacheinander ausgibt.

Schreiben Sie ein Programm, das alle ganzzahligen Werte von $-10°$ bis $+30°$ Celsius nach Fahrenheit umrechnet, die umgerechneten Werte in ein Array speichert und dieses mit `printA` ausgibt.

Aufgabe 7 - Call by Reference

1 Punkt

Schreiben Sie eine Funktion `swap`, welche die Werte von zwei Fließkommazahlen `a` und `b` im Speicher tauscht. Was ist das Problem mit der Funktionssignatur `swap(double a, double b)`? Korrigieren Sie diese und implementieren Sie den Prozedurrumpf.

Verändern Sie das Programm aus Aufgabe 6 so, dass vor der Ausgabe des Array darin jeder Wert mit geradem Index mit dem nächsten Wert mit ungeradem Index mittels `swap` vertauscht wird.

Aufgabe 8 - Speicherverwaltung

1 Punkt

Schreiben Sie eine Funktion `double* c2fArray(int lower, int upper)`, welche mittels `malloc` ein Array vom Typ `double` erstellt und alle umgerechneten Fahrenheit Werte zu den ganzzahligen Celsius Werten von `lower` bis `upper` darin speichert, die Grenzwerte eingeschlossen. Die Funktion soll dieses Array dann (als Pointer) zurückgeben.

Schreiben Sie ein Programm, das mit `c2fArray` ein Array der umgerechneten Werte von $-10°$ bis $+30°$ Celsius erstellt, dieses mit `printA` ausgibt und das Array dann mittels `free` wieder aus dem Speicher freigibt.

Aufgabe 9 - Pointer

1 Punkt

Schreiben Sie eine Funktion `c2fPointer(int *input, float *output)`, welche einen Eingabewert `input` (°C) in einen Ausgabewert `output` (°F) umrechnet, wobei beide Variablen per *call-by-reference*, also als Pointer übergeben werden.

Benutzen Sie `c2fPointer` in dieser `main`-Prozedur:

```
int main() {
    float *out;
    int in = 5;
    c2fPointer(&in, out);
    printf("%d deg C = %f deg F\n", in, *out);
    return 0;
}
```

Was passiert bei der Ausführung und warum? Korrigieren Sie den Fehler ohne `c2fPointer` zu verändern.

Aufgabe 10 - Strings

1 Punkt

Gegeben folgendes Array:

```
int ints[] = {0x31, 0x30, 0x30, 0x25,
              0x20, 0x64, 0x6f, 0x6e,
              0x65, 0x3f, 0x00};
```

Wandeln Sie dies in einen String (Array von `char`-Werten mit abschließendem `0x00` Character) um. Warum ist der Aufruf

```
char *s = (char*) ints;
```

hierfür keine Lösung?

Geben Sie den String auf dem Bildschirm aus. Was ist das Problem mit dem Aufruf `printf(s)`?