Einführung in die Programmierung für Physiker
WS 2017/2018 – Marc Wagner

Klausur

Dauer: 90 Minuten **09. März 2018, 10:15 bis 11:45**

- Alle folgenden Fragen und sämtlicher Programmcode beziehen sich auf die in der Vorlesung ausgiebig diskutierte Programmiersprache C.
- Erlaubte Ausrüstung/Hilfsmittel: Stift, Papier.
- Bücher, Ordner, handgeschriebene Aufzeichnungen, elektronische Geräte sind verboten.
- Bitte beantworte jede Aufgabe auf einer gesonderten DinA4-Seite und gib auf jeder verwendeten Seite Deinen Namen und Matrikelnummer an.
- Bitte gib dieses Deckblatt mit Deinen Aufgaben ab. Die Aufgabenblätter kannst Du behalten.
- Es können insgesamt 40 Punkte erzielt werden.
- Die Klausur ist mit 20 oder mehr Punkten bestanden.

Vollständiger Name: _____

Matrikelnummer: _____

Aufgabe	Punkte
1	/6
2	/4
3	/4
4	/5
5	/4
6	/5
7	/6
8	/6
Summe:	/40

Bestanden: ☐ Ja ☐ Nein

Aufgabe 1 (6 Pkt.)

(i) Beschreibe den Unterschied zwischen einer nicht-konstanten Variable (z.B. vom Typ `int`) und einer konstanten Variable (z.B. vom Typ `const int`). Ist der Programmteil

```
const int N;
N=4;
```

korrekt? Falls nein, warum nicht?

(ii) Beschreibe den Unterschied zwischen den Datentypen `const float*` und `float* const`. Ist der Datentyp `float const*` äquivalent zu einem der beiden zuerst genannten Datentypen? Falls ja, zu welchem?

(iii) Welche Bildschrimausgabe liefert das folgende Programm?

```
#include <stdio.h>

double x=3.14;

int main(void){
    printf("x=%f\n", x);
    {
        double x=2.718;
        printf("x=%f\n", x);
    }
    printf("x=%f\n", x);
}
```

Erläutere anhand des Programms das Konzept von Gültigkeitsbereichen und von Verschattung.

(iv) Welche Bildschrimausgabe liefert das folgende Programm?

```
#include <stdio.h>

void f(void){
    static int N=1;
    printf("Function f called %d time(s).\n", N);
    N++;
}

int main(void){
    f();
    f();
}
```

Welche spezielle Eigenschaft erhält die Variable `N` aufgrund des vorangestellten Schlüsselwortes `static`?

Aufgabe 2 (4 Pkt.)

Schreibe eine Funktion, die als Parameter eine Variable vom Typ `unsigned int` erhält. Der Wert dieser Variable entspricht einer nicht-negativen Anzahl von Tagen. Die Funktion soll diese Anzahl von Tagen in eine äquivalente Anzahl von Jahren, Monaten (0 bis 11) und Tagen (0 bis 29) umwandeln und das Ergebnis in einer den folgenden Beispielen entsprechenden Form am Bildschirm ausgeben. Gehe davon aus, dass jedes Jahr 365 Tage und jeder Monat 30 Tage hat.

```
37 Tag(e) entspricht(entsprechen) 0 Jahr(en), 1 Monat(en), 7 Tag(en).
399 Tag(e) entspricht(entsprechen) 1 Jahr(en), 1 Monat(en), 4 Tag(en).
```

Aufgabe 3 (4 Pkt.)

Gegeben sind folgende Programmzeilen.

```
int a = 12;
double b = 3.0;
```

(i) Welchen Wert liefert der arithmetische Ausdruck `b - 2.0 * 5.0`? Erläutere in diesem Zusammenhang das Konzept des Vorrangs von Operatoren.

(ii) Welchen Wert liefert der arithmetische Ausdruck `12.0 / 2.0 * 3.0`? Erläutere in diesem Zusammenhang das Konzept der Linksassoziativität.

(iii) Welche Werte liefern die folgenden arithmetischen Ausdrücke?

(a) `6 / a + 0.5`

(b) `(double)3 / 4`

(c) `(double)(3 / 4)`

(d) `a++ + 3.0`

Aufgabe 4 (5 Pkt.)

Gegeben sind folgende Programmzeilen.

```
double a[6] = {0.0, 1.0, 2.0, 3.0, 4.0, 5.0};
double *b = &(a[3]);
```

Gib an, welchen Datentyp die folgenden Ausdrücke besitzen. Gib im Fall von einem Zeiger den Inhalt des Speichers an, auf den der Zeiger verweist, d.h. den Wert der dort gespeicherten `double`-Zahl. Gib ansonsten den Wert des Ausdrucks an.

(i) `a[4]`

(ii) `&(a[2])`

(iii) `b[-1]`

(iv) `b+2`

(v) `*b`

Aufgabe 5 (4 Pkt.)

Schreibe ein vollständiges C-Programm, welches die "ersten 1000 Quadratzahlen" n^2, $n = 1, 2, \ldots, 1000$ in eine neuangelegte Textdatei mit Namen `quadratzahlen.txt` schreibt. Die Zahlen sollen dabei in Exponentialschreibweise mit Vorzeichen formatiert sein, d.h. der Inhalt der Textdatei soll wie folgt aussehen.

```
+1.00000e+00
+4.00000e+00
+9.00000e+00
+1.60000e+01
+2.50000e+01
...
+9.96004e+05
+9.98001e+05
+1.00000e+06
```

Aufgabe 6 (5 Pkt.)

Der folgende Programmteil soll zwei 3-elementige `double` Arrays mit den Zahlen `2.0`, `4.0`, `6.0` bzw. `1.0`, `3.0`, `5.0` füllen. Der Speicher für die Arrays soll dabei dynamisch während der Programmlaufzeit angefordert werden. Die beiden Arrays sollen dann als 3-komponentige Vektoren interpretiert, deren Skalarprodukt berechnet und ausgegeben werden.

(i) Welche vier Fehler hat der Programmierer begangen, die unerwünschte und unter Umständen katastrophale Auswirkungen haben können? Erläutere die vier Fehler im Detail und skizziere mögliche Auswirkungen. Gib einen entsprechenden korrigierten Programmteil an.

(ii) Erweitere den Programmcode so, dass der dynamisch reservierte Speicher möglichst frühzeitig wieder freigegeben wird, das Programm aber nach wie vor fehlerfrei arbeitet.

```c
#include <stdio.h>
#include <stdlib.h>
int main(void){
    const int n = 3;
    double *a = (double *)malloc(n);
    a[0] = 2.0;
    a[1] = 4.0;
    a[2] = 6.0;
    double *b = a;
    b[0] = 1.0;
    b[1] = 3.0;
    b[2] = 5.0;
    double sp;
    int j;
    for(j = 1; j <= 3; j++)
        sp += a[j]*b[j];
    printf("a*b = %f\n", sp);
}
```

Aufgabe 7 (6 Pkt.)

Die Multiplikation zweier nicht-negativer ganzer Zahlen m und n kann auf die Addition zurckgeführt werden, indem z.B. n Mal die Zahl m addiert wird. Vervollständige die beiden Funktionen

```c
unsigned int mult_recursive(unsigned int m, unsigned int n){/*...*/}
unsigned int mult_iterative(unsigned int m, unsigned int n){/*...*/}
```

die die Multiplikation der `unsigned int`-Parameter `m` und `n` wie oben beschrieben durchführen sollen (der Multiplikationsoperator `*` darf also nicht verwendet werden). Programmiere eine der beiden Funktionen in rekursiver, die andere in iterativer Art und Weise.

Aufgabe 8 (6 Pkt.)

Definiere eine Struktur `struct c_number`, die zwei `double`-Variablen `re` (entspricht dem Realteil) und `im` (entspricht dem Imaginärteil) enthält. Schreibe drei Funktionen, die als Parameter jeweils zwei Variablen vom Typ `struct c_number` erhalten und die die drei grundlegenden Rechenoperationen

- Addition zweier komplexer Zahlen,
- Subtraktion zweier komplexer Zahlen und
- Multiplikation zweier komplexer Zahlen

ausführen. Die Funktionen sollen das Ergebnis als `struct c_number` zurückliefern.