

Wintersemester 2021/22

## Ingenieurinformatik, Teil 1: C-Programmierung

Schriftliche Fernprüfung mit Videoaufsicht

**Prüfer:** Jäger-Hezel, Küpper, Krug, Ressel, Tasin, Selting, Reichl

**Bearbeitungszeit:** 60 Minuten

**Hilfsmittel:**

- Taschenrechner und elektronische Hilfsmittel sind nicht zugelassen.
- Alle schriftlichen Unterlagen sind erlaubt.
- Der PC darf während der Prüfung nur zur Anzeige des Aufgabenblatts genutzt werden.

Schreiben Sie Ihren Namen, Vornamen und die Studiengruppe auf alle Lösungsblätter. Es werden nur handschriftliche Lösungen auf leeren, weißen DIN-A4-Blättern akzeptiert.

Studierende aus einer alten Studien- und Prüfungsordnung, die zur Kombiprüfung „Ingenieurinformatik“ angemeldet sind, müssen beide Teile (C-Programmierung und Numerik/ Matlab) im selben Semester schreiben.

**\*\*\* Viel Erfolg! \*\*\***

## Aufgabe 1: (ca. 26 Punkte)

Schreiben Sie ein C-Programm, welches eine Nullstelle  $x_0$  der Funktion  $f(x)$  näherungsweise ermittelt.  
Die Funktion  $f(x)$  ist als Unterfunktion zu programmieren:

$$f(x) = \frac{e^{1023 \cdot x} - 1}{e^{511 \cdot x} - 1} - \frac{255}{63}$$

Der Anwender wird zunächst nach zwei x-Werten  $x_1$  und  $x_2$  gefragt, bei denen die Funktionswerte  $f(x_1)$  bzw.  $f(x_2)$  unterschiedliche Vorzeichen haben. Hier ist ein konkretes Beispiel:

$$\begin{aligned}x_1 &= 0.001 & f(x_1) &= -1.37649 < 0 \\x_2 &= 0.003 & f(x_2) &= +1.60218 > 0\end{aligned}$$

Zwischen  $x_1$  und  $x_2$  muss sich also (mindestens) eine Nullstelle  $x_0$  befinden. Dort ist  $f(x_0) = 0$ .

---

Ihr Programm soll die folgenden Schritte ausführen:

1. Es werden zwei x-Werte  $x_1$  und  $x_2$  über die Tastatur eingegeben.
  2. Nach der Eingabe wird folgendermaßen überprüft, ob die eingegebenen Werte  $x_1$  und  $x_2$  gültig sind:
    - 2.1. Wenn  $x_1 \leq 0$  ist oder wenn  $x_2 > 2$  ist, dann wird eine Fehlermeldung auf dem Bildschirm ausgegeben und die Eingabe (Schritt 1) wird wiederholt.
    - 2.2. Die Differenz  $x_2 - x_1$  muss größer als  $10^{-1}$  sein. Wenn dies nicht der Fall ist, dann wird eine Fehlermeldung ausgegeben und die Eingabe (Schritt 1) wird wiederholt.
    - 2.3. Wenn die Funktionswerte  $f(x_1)$  und  $f(x_2)$  dasselbe Vorzeichen haben, dann wird eine Fehlermeldung auf dem Bildschirm ausgegeben und die Eingabe (Schritt 1) wird wiederholt.
  3. Solange  $x_2 - x_1 > 10^{-10}$  ist, werden die folgenden Schritte in einer Schleife immer wieder ausgeführt:
    - 3.1. Der Wert  $x_0 = \frac{x_1+x_2}{2}$  wird berechnet.
    - 3.2. Wenn das Produkt der Funktionswerte  $f(x_0) \cdot f(x_1) > 0$  ist, dann wird der Wert von  $x_0$  in die Variable  $x_1$  geschrieben, sonst wird der Wert von  $x_0$  in die Variable  $x_2$  geschrieben.
  4. Nach dem Ende der Schleife werden  $x_0$  und  $f(x_0)$  mit 8 Nachkommastellen ausgegeben.
  5. Die Anzahl der Schleifendurchläufe (vergl. Schritt 3) wird ebenfalls ausgegeben.
- 

Der Programmablauf soll ähnlich wie in der folgenden Abbildung aussehen:

```
C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
x1 eingeben: 0.001
x2 eingeben: 0.002
Fehler, f(x1) und f(x2) haben dasselbe Vorzeichen.

x1 eingeben: 0.001
x2 eingeben: 0.003
Eingabe durch den Anwender ...

Nullstelle, Näherung:
x0 = 0.00217440 mit f(x0) = 0.00000001
Anzahl Schleifen: 25
```

```
1 import math
2
3 def f(x):
4     return ((math.exp(1023 * x) - 1) / (math.exp(511 * x) - 1) - 255 / 63)
5
6 def find_root(x1, x2):
7     epsilon = 1e-10
8     max_iterations = 1000
9     iteration = 0
10    while abs(x2 - x1) > epsilon and iteration < max_iterations:
11        x0 = (x1 + x2) / 2
12        fx0 = f(x0)
13        if f(x1) * fx0 > 0:
14            x1 = x0
15        else:
16            x2 = x0
17        iteration += 1
18    return x0, iteration
19
20 def main():
21     x1 = float(input("Bitte geben Sie den ersten x-Wert an: "))
22     x2 = float(input("Bitte geben Sie den zweiten x-Wert an: "))
23     while x1 <= 0 or x2 > 2 or x2 - x1 <= 0.001 or f(x1) * f(x2) > 0:
24         if x1 <= 0 or x2 > 2:
25             print("Fehler: x1 muss größer als 0 sein und x2 muss größer oder gleich 2 sein! ")
26         elif x2 - x1 <= 0.1:
27             print("Fehler: x2 muss um 0.001 größer sein als x1")
28         else:
29             print("Fehler: f(x1) und f(x2) müssen verschiedene Vorzeichen haben")
30     x1 = float(input("Bitte geben Sie den ersten x-Wert erneut an:"))
31     x2 = float(input("Bitte geben Sie den zweiten x-Wert erneut an:"))
32     x0, iterations = find_root(x1, x2)
33     print("Nullstellen, Näherung: {:.8f}".format(x0))
34     print("Anzahl der Schleifen: {}".format(iterations))
35
36 if __name__ == '__main__':
37     main()
```

## Aufgabe 2: (ca. 16 Punkte)

Eine globale Matrix ist wie folgt definiert:

```
int m[50][60];
```

- 2.1. Schreiben Sie eine C-Funktion mit dem Namen `fill_m`, welche diese Matrix mit ganzzahligen Zufallszahlen im Bereich von 20 ... 70 füllt (beide Grenzen eingeschlossen).
- 2.2. Entwerfen Sie das Struktogramm für eine Funktion `check_m`. Diese Funktion ermittelt die größte Zahl, die in der globalen Matrix `m` abgespeichert ist. Diese Zahl soll auf dem Bildschirm ausgegeben werden. (Die Rückgabe des Ergebnisses an das Hauptprogramm ist nicht erforderlich.)

Auf dem Bildschirm soll auch ausgegeben werden, wie oft diese Zahl in der Matrix `m` vorkommt.  
(Es ist ja durchaus möglich, dass dieselbe Zahl mehrfach in der Matrix `m` vorkommt.)

Achtung, in diesem Unterpunkt 2.2 soll kein C-Quelltext geschrieben werden!

## Aufgabe 3: (ca. 6 Punkte)

In der Mathematik gibt es die Zahlenfolge der sog. „Dreieckszahlen“: 1, 3, 6, 10, 15, 21, 28, 36, ...

Die  $n$ -te Dreieckszahl ist die Summe aller ganzen Zahlen von 1 bis  $n$ .

Für die fünfte Dreieckszahl gilt also zum Beispiel:  $15 = 1 + 2 + 3 + 4 + 5$

Das folgende C-Programm berechnet die ersten 100 Dreieckszahlen. Im Quelltext befinden sich vier Fehler.

- In welchen Zeilen befinden sich die Fehler?
- Geben Sie jeweils die korrekte Version der kompletten Zeile (!) an.

```
1 #define N 100
2 #include <stdio.h>
3 void dreieck(void);
4 int v(N);
5
6 int main(void)
7 {
8     int i;
9     dreieck;
10    for(i = 0; i < N; ++i) printf("%7d , v[i]\"");
11    return 0;
12 }
13
14 void dreieck(void)
15 {
16     int i, j;
17     for(i = 0; i < N; ++i)
18     {
19         v[i] = 1;
20         for(j = 0; j < i; ++j)
21             v[i] += j + 2;
22     }
23 }
```

```
def Dreieckszahlen(n):
    Dreiecksnummer = []
    for i in range(1, n + 1):
        Dreiecksnummer.append(sum(range(1, i + 1)))
    return Dreiecksnummer

print(Dreieckszahlen(100))
```

2.1

2.2

```
import random      ...
row = 1
col = 1

x = 50
y = 60

z = 0
r= 0
maxZahl = 0
zähler = 0

m = [[0] * x for _ in range(y)]

for row in range(y):
    for col in range(x):
        m[row][col] = random.randint(20, 70)

print(m)

for row in range(y):
    for col in range(x):
        z = m[row][col]
        if z > maxZahl:
            maxZahl = z

for row in range(y):
    for col in range(x):
        r = m[row][col]
        if r == maxZahl:
            zähler += 1

print(maxZahl)
print(zähler)
```

#### Aufgabe 4: (ca. 5 Punkte)

Wie lautet die Ausgabe der Funktion test(), wenn Sie in dem folgenden Programm den Namen **Zarah** eingeben?

```
#include <stdio.h>
void test(char *x);

int main(void)
{
    char str[100];
    printf("Namen: ");
    scanf("%99s", str);
    test(str);
    return 0;
}

void test(char *x)
{
    int i = 0;
    while(x[i] != 0)
    {
        printf("%c %c\n", x[i], x[i] + 1);
        ++i;
    }
}
```

Ausschnitt aus der ASCII-Tabelle:

60 <	61 =	62 >	63 ?
64 @	65 A	66 B	67 C
68 D	69 E	70 F	71 G
72 H	73 I	74 J	75 K
76 L	77 M	78 N	79 O
80 P	81 Q	82 R	83 S
84 T	85 U	86 V	87 W
88 X	89 Y	90 Z	91 [
92 \	93 ]	94 ^	95 _
96 `	97 a	98 b	99 c
100 d	101 e	102 f	103 g
104 h	105 i	106 j	107 k
108 l	109 m	110 n	111 o
112 p	113 q	114 r	115 s
116 t	117 u	118 v	119 w
120 x	121 y	122 z	123 {
124	125 }	126 ~	127 }

#### Aufgabe 5: (ca. 14 Punkte)

5.1. Sie arbeiten mit einem aktuellen C-Compiler auf einem Windows- oder macOS-Rechner.

- Wie viele Werte (Elemente) sind in den folgenden Vektoren und Matrizen gespeichert?
- Wie viel Speicher wird von diesen Vektoren bzw. Matrizen im Hauptspeicher des Rechners benötigt?

$$4 \cdot 3 \cdot 2 \cdot 1 = 24 \text{ Elemente/Werte gespeichert}$$

float x[4][3][2][1];  
int x[1][2][3][4];  
char name[101];

- a) Anzahl Elemente?  
c) Anzahl Elemente?  
e) Anzahl Elemente?

- b) Speicherplatz-Bedarf in Bytes?  
d) Speicherplatz-Bedarf in Bytes?  
f) Speicherplatz-Bedarf in Bytes?

für einen float wird  
wegen 4 Bytes  
benötigt somit  
 $4 \cdot 24 = 96 \text{ Bytes}$   
96 Bytes

5.2. Der Anwender gibt die drei Zeichen „{}“ mittels `scanf("%100s", name);` in die Variable `name` ein (siehe Unterpunkt 5.1). Direkt nach dem letzten Zeichen wird die Eingabetaste gedrückt. Welcher Wert steht anschließend im Element `name[3]`?

5.3. Wie lautet die Ausgabe des folgenden Programms?

```
#include <stdio.h>
int main(void)
{
    double d1;
    int z1 = 11, z2 = 4, z3;
    z3 = z1 % z2; printf("%d\n", z3);
    d1 = z1 / z2; printf("%.1f\n", d1);
    return 0;
}
```

5.4. Unterscheidet sich der Speicherplatz-Bedarf der Variablen `d` und `s`? (Kurze Begründung!)

double \*d; → double verbraucht mehr Speicherplatz als eine der Variablen  
char \*s;

5.5. Ein Student fragt seinen Informatiklehrer nach einem Beispielprogramm mit einer „if-Schleife“. Der Informatiklehrer ärgert sich ...

Erläutern Sie, was an dem Begriff „if-Schleife“ so problematisch ist.



Sommersemester 2021

## Ingenieurinformatik, Teil 1: C-Programmierung

Schriftliche Fernprüfung mit Videoaufsicht

**Prüfer: Selting, Ressel, Küpper, Reichl und KollegInnen**

**Bearbeitungszeit: 60 Minuten**

**Hilfsmittel:**

- Taschenrechner und elektronische Hilfsmittel sind nicht zugelassen.
- Alle schriftlichen Unterlagen sind erlaubt.
- Der PC darf während der Prüfung nur zur Anzeige des Aufgabenblatts genutzt werden.

**Schreiben Sie Ihren Namen, Vornamen und auch die Studiengruppe auf alle Lösungsblätter. Es werden nur handschriftliche Lösungen auf leeren, weißen DIN-A4-Blättern akzeptiert.**

**Wenn Sie zur Kombiprüfung „Ingenieurinformatik“ angemeldet sind, dann beachten Sie bitte, dass Sie beide Teile (C-Programmierung und Numerik/ Matlab) im selben Semester schreiben müssen.**

**\*\*\* Viel Erfolg! \*\*\***

### Aufgabe 1 (allg. Programmierung, ca. 24 Punkte)

Eine Masse wird unter dem Winkel  $\varphi$  zur x-Achse mit einer Anfangsgeschwindigkeit

$$\vec{v}_0 = \begin{pmatrix} v_{x,0} \\ v_{y,0} \end{pmatrix} = \begin{pmatrix} 5 \cos(\varphi) \\ 5 \sin(\varphi) \end{pmatrix} \left[ \frac{m}{s} \right]$$

aus einer Höhe  $h = 10$  Meter abgeworfen.

Beim sog. schießen Wurf ohne Reibung gelten für die zeitabhängigen Koordinaten  $x(t)$ ,  $y(t)$  des Massenschwerpunktes die Gleichungen

$$x(t) = v_{x,0} \cdot t$$

$$y(t) = h + v_{y,0} \cdot t - \frac{1}{2} \cdot g \cdot t^2 \quad \text{mit der Erdbeschleunigung } g = 9,81 \left[ \frac{m}{s^2} \right]$$

Die Abbildung oben zeigt die Masse (rote Kugel) während des Wurfs zu verschiedenen Zeitpunkten im x-y-Koordinatensystem. Die Masse kommt zum Zeitpunkt  $t_{End}$  auf dem Boden auf d.h.  $y(t)=0$ :

$$t_{End} = \frac{v_{y,0}}{g} + \sqrt{\left( \frac{v_{y,0}}{g} \right)^2 + \frac{2h}{g}} \quad [s]$$

**Schreiben Sie ein C-Programm, welches die folgenden Aufgaben löst:**

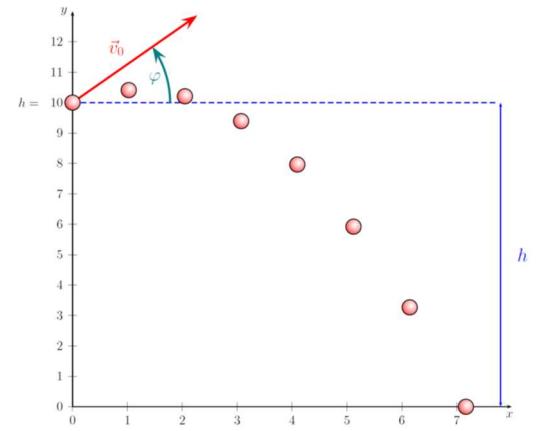
- Beim Start wird der Anwender nach dem Abwurf-Winkel  $\varphi$  im **Gradmaß** gefragt, mit der Vorgabe  $10^\circ \leq \varphi \leq 35^\circ$ . Falls die Winkel-Eingabe nicht im Bereich zwischen 10 und 35 Grad liegt, soll eine Fehlermeldung ausgegeben und die Eingabe wiederholt werden, bis ein Winkel aus diesem Bereich eingegeben wurde.
- Das Zeitintervall  $[0, t_{End}]$  soll in 30 Teilintervalle  $[t_i, t_{i+1}]$  aufgeteilt werden  
 $t_0 = 0, t_1, t_2, \dots, t_{30} = t_{End}$   
 und die Zeitpunkte  $t_i$  (in Sekunden) sowie  $x(t_i)$  und  $y(t_i)$  (in Metern) sollen jeweils mit **vier Nachkommastellen** tabellarisch ausgegeben werden (siehe nachfolgende Abbildung).
- Kurz vor dem Ende des Programms, nachdem (!) die Tabelle vollständig auf dem Bildschirm ausgegeben wurde, werden die folgenden Informationen ausgegeben:
  1. Die maximale Höhe  $h_{max}$ , die bei  $t_{max} \in \{t_0, t_1, \dots, t_{30}\}$  erreicht wurde und auch der Zeitpunkt  $t_{max}$  (jeweils **zwei Nachkommastellen**).
  2. Die zur maximalen Höhe  $h_{max}$  gehörigen x- und y-Koordinaten  $x_{max} = x(t_{max})$  und  $y_{max} = y(t_{max})$  (jeweils **zwei Nachkommastellen**).

**Die Bildschirmausgabe Ihres Programms soll dem folgenden Bildschirmfoto entsprechen:**

```
C:\Qt\Qt5.14.2\Tools\QtCreator\bin\qtcreator_process_stub.exe
Geben Sie den Winkel phi in Grad ein ( 10<=phi<=35 ): 40
Falsche Eingabe fuer phi: 10<=phi<=35
Geben Sie den Winkel phi in Grad ein ( 10<=phi<=35 ): 30 Benutzereingabe

t          x(t)        y(t)
-----
0.0000    0.0000    10.0000
0.0568    0.2461    10.1263
0.1137    0.4923    10.2208
0.1705    0.7384    10.2837
...
1.5347    6.6456    2.2837
1.5916    6.8917    1.5541
1.6484    7.1378    0.7929
1.7052    7.3839    0.0000

Die maximale Hoehe 10.28 Meter wird nach 0.17 Sekunden erreicht.
Die zugehoerigen Koordinaten sind xmax= 0.74 und ymax=10.28
```



```

60
61 import math
62
63 g = 9.81
64 h = 10
65
66 while True:
67     phi = float(input("Geben Sie bitte den Winkel phi in Grad an ( 10 <= phi <= 35): "))
68     if 10 <= phi <= 35:
69         break
70     else:
71         print("Falscher Winkel eingegeben. Versuchen Sie es erneut, der Winkel soll zwischen 10 und 35 Grad liegen.")
72
73 phi = math.radians(phi)
74 vx0 = 5 * math.cos(phi)
75 vy0 = 5 * math.sin(phi)
76
77 t_end = ((vy0 / g) + math.sqrt((vy0 / g) ** 2 + 2 * h / g))
78 dt = t_end / 30
79
80 t = 0
81 max_h = 0
82 max_t = 0
83 max_x = 0
84 max_y = 0
85
86 print("t\tx(t)\ty(t)")
87 print("----\t----\t----")
88
89 for i in range(31):
90     x = vx0 * t
91     y = h + vy0 * t - 0.5 * g * t ** 2
92
93     if y > max_h:
94         max_h = y
95         max_t = t
96         max_x = x
97         max_y = y
98
99     print("{:.4f}\t{:.4f}\t{:.4f}".format(t, x, y))
100
101 print("\nDie maximale Höhe beträgt: {:.2f} Die Höhe wird nach t = {:.2f} Sekunden erreicht".format(max_h, max_t))
102

```

## Aufgabe 2 (Struktogramm, ca. 9 Punkte)

In einem C-Programm sind die Matrix (Preistabelle: `pt`) sowie der Preis-Vektor (Einzelpreis: `ep`) als globale Variablen definiert:

```
double pt[10][4], ep[10];
```

Ein Schraubenhersteller verkauft 10 verschiedene Typen von Schrauben (Typ 1-10).

Der Einzelpreis für eine Packung Schrauben vom Typ k steht als k-tes Element im Vektor `ep`.

Für Bestellungen bis 5 Packungen eines Typs gilt der jeweilige Einzelpreis. Bei einer Bestellung von 6 oder mehr Packungen wird ein Rabatt pro Packung wie in untenstehender Tabelle gewährt:

Packungsanzahl	1-5	6-10	11-20	>20
Rabatt für alle Preiskategorien	0%	3%	7%	15%

Es soll eine Funktion mit dem Namen „belegen“ programmiert werden. Diese Funktion schreibt die folgenden Werte in die Matrix „`pt`“:

- In der 1. Spalte steht für Bestellungen bis 5 Packungen der Einzelpreis (=Vektor „`ep`“),
- in der 2. Spalte steht für Bestellungen von 6 bis 10 Packungen der rabattierte Preis,
- in der 3. Spalte steht für Bestellungen von 11 bis 20 Packungen der rabattierte Preis,
- in der letzten Spalte steht für Bestellungen über 20 Packungen der rabattierte Preis

In jeder Spalte steht der (rabattierte) Preis für jeden Typ 1-10.<sup>1</sup>

Typ	0-5 Stück	6-10 Stück	11-20 Stück	ab 21 Stück
1	1.25	1.21	1.16	1.06
2	1.70	1.65	1.58	1.44
⋮	⋮	⋮	⋮	⋮
10	3.25	3.15	3.02	2.76

Zeichnen Sie ein Struktogramm der Funktion „belegen“. Alle Kontrollstrukturen (**2 verschachtelte Schleifen, 1 Mehrfachverzweigung**) müssen klar erkennbar und korrekt gezeichnet sein.

**Hinweis: In dieser Aufgabe 2 soll kein C-Quelltext geschrieben werden!**

<sup>1</sup> Die Preise in der Ausgabe der Beispieldatei `pt` sind auf 2 Nachkommastellen gerundet.

### **Aufgabe 3 (Funktionen, Felder, ca. 10 Punkte)**

In einem C-Programm sind alle Variablen als lokale Variablen definiert. Es soll eine Funktion „**skalarprod**“ mit der Deklaration

```
double skalarprod(double *x, double *y, int n);
```

programmiert werden, welche das Skalarprodukt  $\langle \vec{x}, \vec{y} \rangle$  der Vektoren  $\vec{x}, \vec{y} \in \mathbb{R}^n$  berechnet.

In einer Formelsammlung finden Sie z. B.:

$$\langle \vec{x}, \vec{y} \rangle := \sum_i x_i y_i$$

- a) Schreiben Sie den C-Quelltext der Funktion „**skalarprod**“. Diese Funktion soll das Skalarprodukt zweier beliebiger Vektoren  $\vec{x}, \vec{y} \in \mathbb{R}^n$  in einer Schleife berechnen und das Ergebnis als Rückgabewert an das aufrufende Programm liefern.

**Hinweis: In dieser Teilaufgabe 3a) soll kein Struktogramm gezeichnet werden!**

- b) Wie lautet der Funktionsaufruf, in der unten fehlenden Zeile, wenn hier mit Hilfe der Funktion „**skalarprod**“ aus Teilaufgabe 3a) der Wert der Variablen **norm2** gemäß der Formel

$$\text{norm2} = \langle \vec{x}, \vec{x} \rangle$$

berechnet werden soll.

```
1  #include <stdio.h>
2  #include <math.h>
3  double skalarprod(double *x, double *y, int n);
4
5  int main(void)
6  {
7      double x[] = {1.3, 2.0, 3.7, 4.1, 5.9, 6.1, 7.6, 8.8, 9.3, 10.0};
8      double norm2;
9
10
11      Hier kommt der Funktionsaufruf von „skalarprod“
12
13
14      :
15 }
```

#### Aufgabe 4 (Fehlersuche, ca. 10 Punkte)

Das folgende Programm belegt zwei Vektoren mit Zufallszahlen aus dem Bereich [5.0, 8.0] und berechnet anschließend das aus der Mathematik bekannte Vektor- oder Kreuzprodukt.

In einer Formelsammlung finden Sie z. B.:

$$\vec{u} \times \vec{v} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \times \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{pmatrix}$$

Im Quelltext befinden sich **fünf Fehler**.

- In welchen Zeilen befinden sich die Fehler?
- Geben Sie jeweils die korrekte Version der kompletten (!) Zeile an.

```
1 #include <stdio.h>
2
3 #define DIM 3
4
5 int main()
6 {
7     float x[DIM], y[DIM], erg[DIM];
8     int i = 0;
9     while(i <= DIM)
10    {
11        x[i] = 3.0*rand() / RAND_MAX + 8.0;
12        y[i] = 3.0*rand() / RAND_MAX + 5.0;
13    }
14 }
15 //
16 printf("Die Zufallszahlen liegen ");
17 printf("im Intervall [%d, %d]\n", 3, 8);
18 //
19 erg[0] = x[1]*y[2]-x[2]*y[1];
20 erg[1] = x[2]*y[0]-x[0]*y[3];
21 erg[2] = x[0]*y[1]-x[1]*y[0];
22 //
23 printf("%9.4f      %9.4f      %9.4f\n", x[0], y[0], erg[0]);
24 printf("%9.4f  x  %9.4f  =  %9.4f\n", x[1], y[1], erg[1]);
25 printf("%9.4f      %9.4f      %9.4f\n\n", x[2], y[2], erg[2]);
26
27 }
```

## Aufgabe 5 (Operatoren, Bytes, Zeichenketten, ca. 14 Punkte)

4.1. Beantworten Sie zu folgendem Quelltext-Ausschnitt die Fragen a) – d)

1 Byte  
pro Element

```
5     char text[11] = "SoSe2021";
6     printf("%s \n", text); → print(text, end='|n')
7     text[6] = '\0';
8     printf("%s \n", text);
9     printf("%d Zeichen \n", (int) strlen(text));
```

- a) Wieviel Byte Speicherplatz benötigt die Variable text ? 11 Byte
- b) Welche Programmausgabe erzeugt das obige Quelltext-Teilstück in Zeile 6 ?
- c) Welche Programmausgabe erzeugt das obige Quelltext-Teilstück in Zeile 8 ?
- d) Welche Programmausgabe erzeugt das obige Quelltext-Teilstück in Zeile 9 ?

4.2. Welche Werte haben die Variablen v und w nach Ausführen der nachfolgenden Quelltextzeilen:

```
11     int x,y,v,w;
12     x = 2; y = 3;
13     v = x++ + y++;
14     w = ++x + ++y;
```

$$v = 5 \\ w = 7$$

```
x = 2
y = 3

v = x + y
x += 1
y += 1

w = x + y
```

Begründen Sie die Berechnung Ihrer Werte.

4.3. Zur 2-Byte short Variablen i wird der dezimale Wert 3 addiert:

```
short i = 32767;
i = i + 3;      32770
```

Welchen Wert hat die Variable i nach der Addition? Begründen Sie Ihre Antwort.  
(Hinweis:  $32767 = 2^{15}-1$ )

4.4. Unterscheidet sich der Speicherplatzbedarf der Variablen x und y wenn diese wie folgt definiert sind:

↗ normalweise ↗ Bytes Speicherplatz  
float \*x;  
double \*y;

Begründen Sie Ihre Antwort.

↘ 8 Bytes Speicherplatz



Wintersemester 2020/21

## Ingenieurinformatik, Teil 1: C-Programmierung

Schriftliche Fernprüfung mit Videoaufsicht

**Prüfer: Küpper, Reichl und KollegInnen**

**Bearbeitungszeit: 60 Minuten**

**Hilfsmittel:**

- Taschenrechner und elektronische Hilfsmittel sind nicht zugelassen.
- Alle schriftlichen Unterlagen sind erlaubt.
- Der PC darf während der Prüfung nur zur Anzeige des Aufgabenblatts genutzt werden.

**Schreiben Sie Ihren Namen, Vornamen und auch die Studiengruppe auf alle Lösungsblätter. Es werden nur handschriftliche Lösungen auf leeren, weißen DIN-A4-Blättern akzeptiert.**

**Wenn Sie zur Kombiprüfung „Ingenieurinformatik“ angemeldet sind, dann beachten Sie bitte, dass Sie beide Teile (C-Programmierung und Numerik/ Matlab) im selben Semester schreiben müssen.**

**\*\*\* *Viel Erfolg!* \*\*\***

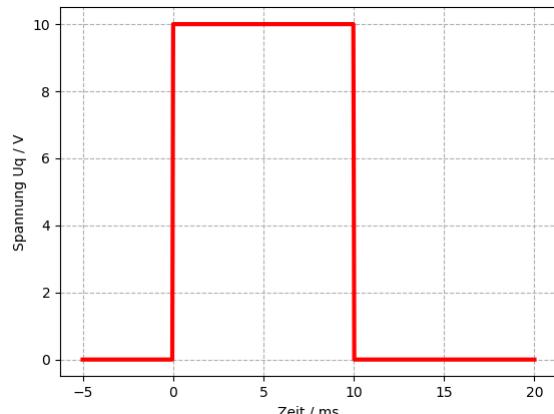
### Aufgabe 1 (allg. Programmierung, ca. 24 Punkte)

Eine Reihenschaltung aus einem Widerstand der Größe  $R = 10\Omega$  und einer Spule  $L = 0,1\text{H}$  ist an eine Spannungsquelle  $u_q(t)$  angeschlossen. Die Spannungsquelle  $u_q(t)$  wird nur zu Beginn kurz eingeschaltet (siehe Abbildung rechts).

Die Stromstärke  $i(t)$  kann folgendermaßen berechnet werden:

$$i(t) = 1 \text{ A} \cdot \left(1 - e^{-\frac{t}{0,01 \text{ s}}}\right) \quad \text{für } t = 0 \dots 0,01 \text{ s}$$

$$i(t) = 1 \text{ A} \cdot \left(1 - \frac{1}{e}\right) \cdot e^{-\frac{t-0,01 \text{ s}}{0,01 \text{ s}}} \quad \text{für } t > 0,01 \text{ s}$$



Schreiben Sie ein C-Programm, welches die folgenden Aufgaben löst:

- Beim Start wird der Anwender nach einem Schwellenwert  $i_1$  für die Stromstärke gefragt. Die Eingabe soll in Ampere (A) erfolgen.
- Die Stromstärke  $i(t)$  wird für das Zeitintervall  $t = 0 \dots 0,02 \text{ s}$  in Schritten von  $\Delta t = 0,0001 \text{ s}$  berechnet. Die Zeit in s und die Stromstärke in A sollen jeweils mit vier Nachkommastellen tabellarisch ausgegeben werden.
- Kurz vor dem Ende des Programms, nachdem (!) die Tabelle vollständig auf dem Bildschirm ausgegeben wurde, werden die folgenden Informationen ausgegeben:
  1. Die maximale Stromstärke  $i_{\max}$ , die im Zeitintervall  $t = 0 \dots 0,02 \text{ s}$  geflossen ist und auch der Zeitpunkt  $t_{\max}$ , wann diese maximale Stromstärke erreicht wurde (jeweils vier Nachkommastellen).
  2. Der Zeitpunkt  $t_1$ , wann die Stromstärke zum ersten Mal den Schwellenwert  $i_1$  erreicht hat (vier Nachkommastellen, Abbildung unten links). Es kann auch sein, dass der Schwellenwert gar nicht erreicht wird; in diesem Fall soll eine entsprechende Meldung ausgegeben werden (Abbildung unten rechts).

Die Bildschirmausgabe Ihres Programms soll den folgenden Bildschirmfotos entsprechen:

<pre>C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe i1 in A: 0.5 Eingabe des Anwenders t = 0.0000 s, i = 0.0000 A t = 0.0001 s, i = 0.0100 A t = 0.0002 s, i = 0.0198 A t = 0.0003 s, i = 0.0296 A t = 0.0004 s, i = 0.0392 A t = 0.0005 s, i = 0.0488 A t = 0.0198 s, i = 0.2396 A t = 0.0198 s, i = 0.2372 A t = 0.0199 s, i = 0.2349 A t = 0.0200 s, i = 0.2325 A tmax = 0.0199 s, imax = 0.2349 A i1 wird zum Zeitpunkt t1 = 0.0070 s erreicht.</pre>	<pre>C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe i1 in A: 0.7 t = 0.0000 s, i = 0.0000 A t = 0.0001 s, i = 0.0100 A t = 0.0002 s, i = 0.0198 A t = 0.0003 s, i = 0.0296 A t = 0.0004 s, i = 0.0392 A t = 0.0005 s, i = 0.0488 A t = 0.0198 s, i = 0.2396 A t = 0.0198 s, i = 0.2372 A t = 0.0199 s, i = 0.2349 A t = 0.0200 s, i = 0.2325 A tmax = 0.0199 s, imax = 0.2349 A i1 wird nicht erreicht!</pre>
---	---

• Vinc übung.py > ...

• • •

```
1 import math as m
2
3 schwellwert = float(input("Stromstärke in Ampere (A): "))
4 t = 0
5 time = 0
6 t1 = 0
7 iMax = 0
8 tMax = 0
9 t0 = 0
10 t1Erreicht = False
11
12 def stromstärke(t):
13     if t <= 0.01:
14         x = float(schwellwert * (1 - m.exp( (-1) * t/0.01)))
15         print(f"t = {t:.4f} s, i = {x:.4f} A")
16         #print(x)
17     elif t > 0.01:
18         y = float(schwellwert * (1 - (1 / m.e)) * m.exp((t-0.01)/0.01))
19         print(f"t = {t:.4f} s, i = {y:.4f} A")
20
21 def stromMax(t1):
22     if t1 <= 0.01:
23         return float(schwellwert * (1 - m.exp( (-1) * t1/0.01)))
24
25     elif t1 > 0.01:
26         return float(schwellwert * (1 - (1 / m.e)) * m.exp((t1-0.01)/0.01))
27
28
29 while time <= 0.02:
30     stromstärke(time)
31
32     x = stromMax(time)
33
34     if x > iMax:
35         iMax = x
36         tMax = time
37
38     if stromMax(time) > schwellwert and False == t1Erreicht:
39         t0 = time
40         t1Erreicht = True
41
42     time += 0.0001
43
44     #print(iMax)
45     #print(tMax)
46
47 print(f"tmax = {tMax:.4f} s , imax = {iMax:.4f} A")
48
49
50
51
52 if t0 == 0:
53     print("i1 wird nicht erreicht!")
54 else:
55     print(f"i1 wird zum zeitpunkt t1 = {t0:.4f} s erreicht")
```

## **Aufgabe 2 (Vektoren, Matrizen, Kontrollstrukturen, ca. 16 Punkte)**

- 2.1. In einem C-Programm ist die folgende Matrix als globale Variable definiert:

```
double m[10][10];
```

Es soll eine Funktion mit dem Namen „belegen“ programmiert werden. Diese Funktion schreibt die folgenden Werte in die Matrix: Alle Elemente „auf dem Rand“ der Matrix (also in der ersten und letzten Zeile bzw. in der ersten und letzten Spalte) sollen auf 10 gesetzt werden. Die übrigen Elemente „im Inneren“ der Matrix sollen auf -1 gesetzt werden.

10	10	10	10	10	10	10	10	10	10
10	-1	-1	-1	-1	-1	-1	-1	-1	10
10	-1	-1	-1	-1	-1	-1	-1	-1	10
10	-1	-1	-1	-1	-1	-1	-1	-1	10
10	-1	-1	-1	-1	-1	-1	-1	-1	10
10	-1	-1	-1	-1	-1	-1	-1	-1	10
10	-1	-1	-1	-1	-1	-1	-1	-1	10
10	-1	-1	-1	-1	-1	-1	-1	-1	10
10	-1	-1	-1	-1	-1	-1	-1	-1	10
10	10	10	10	10	10	10	10	10	10

Zeichnen Sie ein Struktogramm der Funktion „belegen“. Alle verwendeten Kontrollstrukturen müssen klar erkennbar und korrekt gezeichnet sein.

**Hinweis: In dieser Teilaufgabe 2.1 soll kein C-Quelltext geschrieben werden!**

- 2.2. In einem C-Programm ist der folgende Vektor als globale Variable definiert:

```
double v[45];
```

Es soll eine Funktion „fib“ programmiert werden, welche alle (!) Elemente des Vektors der Reihe nach mit den folgenden Werten belegt:

1 1 2 3 5 8 13 21 34 55 89 ...

Es handelt sich dabei um die sog. Fibonacci-Zahlenfolge. Dabei ergibt immer die Summe zweier aufeinanderfolgender Zahlen die nachfolgende Zahl.

Schreiben Sie den C-Quelltext der Funktion „fib“. Diese Funktion soll die Fibonacci-Zahlen in einer Schleife berechnen und in den Vektor schreiben. (Rekursive Lösungen sind nicht zulässig!)

**Hinweis: In dieser Teilaufgabe 2.2 soll kein Struktogramm gezeichnet werden!**

### Aufgabe 3 (Fehlersuche, Zeiger, ca. 13 Punkte)

Nach dem Programmstart gibt der Anwender zunächst alle 10 Elemente eines double-Vektors ein. Anschließend wird die Funktion „summ\_prod“ aufgerufen. Diese Funktion berechnet sowohl die Summe als auch das Produkt von allen Elementen des Vektors.

```
1 #include <stdio.h>
2 #include <math.h>
3
4 Die Deklaration der Funktion „summ_prod“ fehlt noch ...
5
6
7 int main(void)
8 {
9     double vek[10], summ, prod;
10
11    int anz = 0;
12    while(anz < 10);
13    {
14        printf("Element %d: ", anz + 1);
15        scanf("%f", &vek[anz]);
16    }
17
18    summ_prod(vek, anz, &summ, &prod);
19    printf("Summe: %.1f\n", summ);
20    printf("Produkt: %.1f\n", prod);
21    return 0;
22}
23
24
25 Die erste Zeile der Funktions-Definition fehlt noch ...
26
27 {
28     int n;
29     *summ = 0;
30     *prod = 0;
31
32     for(n = 0; n < anz; n = n++)
33     {
34         *summ += v[n];
35         *prod *= v[n];
36     }
37 }
```

3.1. Die Deklaration und die erste Zeile der Funktions-Definition von „summ\_prod“ fehlen noch.

- Wie lautet die Deklaration der Funktion „summ\_prod“?
- Wie lautet die erste Zeile der Funktions-Definition von „summ\_prod“?

3.2. Im Quelltext befinden sich fünf Fehler.

- In welchen Zeilen befinden sich die Fehler?
- Geben Sie jeweils die korrekte Version der kompletten (!) Zeile an.

#### Aufgabe 4 (Bits und Bytes, Zeichenketten, ca. 14 Punkte)

- 4.1. Sie arbeiten mit einem aktuellen C-Compiler auf einem Windows- oder macOS-Rechner.
- Wie viele Werte (Elemente) sind in den folgenden Vektoren und Matrizen gespeichert?
  - Welcher Speicherplatz wird von diesen Vektoren bzw. von diesen Matrizen im Hauptspeicher des Rechners benötigt?
- |                                    |                     |                             |
|------------------------------------|---------------------|-----------------------------|
| <i>400 Elemente</i>                | <i>* 8</i>          | <i>3200 Bytes</i>           |
| <code>double m1[10][20][2];</code> | a) Anzahl Elemente? | b) Speicherbedarf in Bytes? |
| <i>24 Elemente</i>                 |                     | <i>4 * 24 = 96 Bytes</i>    |
| <code>int v[1][2][3][4];</code>    | c) Anzahl Elemente? | d) Speicherbedarf in Bytes? |
| <code>char my_text[20];</code>     | e) Anzahl Elemente? | f) Speicherbedarf in Bytes? |
|                                    | <i>20 Elemente</i>  | <i>20 Bytes</i>             |
- 4.2. Der Anwender gibt die drei Zeichen „abc“ mit `scanf("%19s", my_text);` in die oben definierte Variable `my_text` ein. Direkt nach dem letzten Zeichen wird die Eingabetaste gedrückt. Welcher Wert steht anschließend im Element `my_text[3]`?
- 4.3. Warum ist es eine schlechte Idee, eine Zeichenkette mit dem Befehl `scanf("%s", my_text);` eingeben zu lassen? (Kurze Begründung)
- 4.4. Zur Verarbeitung von Kommazahlen gibt es die Datentypen `float` und `double`. Wann sollte man den Datentyp `double` verwenden? Geben Sie ein Beispiel bzw. einen Grund an.
- 4.5. Wann sollte man anstelle von `double` besser den Datentyp `float` verwenden? Geben Sie ein Beispiel bzw. einen Grund an.



# Ingenieurinformatik

## C-Programmierung

Name	Vorname	Semestergruppe	Hörsaal

	Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5	Summe

**Beginn Ihres Bachelorstudiums:**

- vor WS13/14 (Kombiprüfung C + MATLAB) \*\*
- von WS13/14 bis WS15/16 \*\*
- von SS16 bis WS17/18 (Kombiprüfung C + MATLAB)
- ab SS18

**\*\* Die Prüfung ist nur dann gültig, wenn Sie die Zulassungsvoraussetzung erworben haben (erfolgreiche Teilnahme am Praktikum).**

**Aufgabensteller:** Dr. Reichl, Dr. Küpper und Kollegen

**Bearbeitungszeit:** 60 Minuten

**Hilfsmittel:** Taschenrechner nicht zugelassen,  
PC/Notebook nicht zugelassen,  
sonstige eigene Hilfsmittel sind erlaubt,  
Bearbeitung mit Bleistift ist erlaubt.

**\*\*\* Viel Erfolg! \*\*\***

### Aufgabe 1: (ca. 23 Punkte)

Schreiben Sie ein C-Programm, welches die Geschwindigkeit einer Saturn-V-Rakete während der Brenndauer der ersten Raketenstufe in Zeitschritten der Größe  $\Delta t = 0,05$  s näherungsweise berechnet. Die Ergebnisse sollen tabellarisch auf dem Bildschirm ausgegeben werden.

Folgende Daten sind gegeben:

- Startmasse der Rakete (inkl. Treibstoff und Nutzlast):  $m = 2,75 \cdot 10^6$  kg  
(Den größten Teil der Startmasse, nämlich 72,7%, macht der Treibstoff der ersten Stufe aus!)
- Treibstoffverbrauch pro Sekunde („Brennrate“):  $R = 13,5 \cdot 10^3$  kg/s
- Schubkraft der ersten Raketenstufe:  $F_{\text{Schub}} = 35,1 \cdot 10^6$  N
- Fallbeschleunigung:  $g = 9,81$  m/s $^2$

- Zu Beginn ( $t = 0$ ) befindet sich die Rakete in Ruhe ( $v = 0$ ).
- Wenn zu einem beliebigen Zeitpunkt  $t$  die Masse  $m(t)$  und die Geschwindigkeit  $v(t)$  bekannt sind, können  $m(t + \Delta t)$  und  $v(t + \Delta t)$  zum Zeitpunkt  $t + \Delta t$  wie folgt berechnet werden:

$$v(t + \Delta t) = v(t) + \Delta t \cdot a(t) \quad \text{mit der Beschleunigung } a(t) = \frac{F_{\text{Schub}}}{m(t)} - g$$
$$m(t + \Delta t) = m(t) - R \cdot \Delta t$$

- Nach jedem Zeitschritt  $\Delta t = 0,05$  s werden die aktuellen Werte von  $t$  (zwei Nachkommastellen) und  $v$  (drei Nachkommastellen) tabellarisch ausgegeben.
- Die Berechnungsschleife wird beendet, wenn der Treibstoff der ersten Stufe zu mehr als 95% aufgebraucht ist.
- Unmittelbar vor dem Programmende werden die folgenden drei Werte mit drei Nachkommastellen ausgegeben:
  - Die Beschleunigung  $a$  beim Start der ersten Raketenstufe (zum Zeitpunkt  $t = 0$ ),
  - die Beschleunigung  $a$  unmittelbar vor dem Ende der Berechnungsschleife,
  - der Zeitpunkt  $t$ , wann die Rakete erstmals die Schallgeschwindigkeit von 340 m/s überschreitet.

Die Ausgabe Ihres C-Programms soll so aussehen:

```
C:\Qt\Tools\QtCreator\bin\q... - X
t = 0.05 s, v = 0.148 m/s
t = 0.10 s, v = 0.296 m/s
t = 0.15 s, v = 0.444 m/s
t = 0.20 s, v = 0.592 m/s
t = 0.25 s, v = 0.740 m/s
t = 0.30 s, v = 0.888 m/s
t = 140.55 s, v =
t = 140.60 s, v = 1666.909 m/s
t = 140.65 s, v = 1668.479 m/s
t = 140.70 s, v = 1670.050 m/s

Anfangsbeschleunigung: 2.954 m/s^2
Endbeschleunigung: 31.425 m/s^2
Schallgeschwindigkeit bei t = 62.600 s
```



*Lösung Versuch 6*

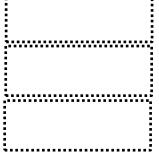
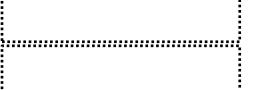
### Aufgabe 2: (ca. 22 Punkte)

Das folgende C-Programm prüft, ob ein Vektor sortiert ist (also ob die Werte im Vektor monoton steigend oder monoton fallend sind) oder nicht.

```
#include <stdio.h>
#include <stdlib.h>

#define DIM 5
double vect[DIM];
void fill_random(void);
int test_monoton(void);

int main(void)
{
    int test;
    fill_random();
    test = test_monoton();

    switch(test)
    {
        case  : printf("Monoton steigend\n");
        case  : printf("Monoton fallend\n");
        case  : printf("Nicht sortiert\n");
    }
    return 0;
}

int test_monoton(void)
{
    int idx = 1, steigt = 0, faellt = 0, ret = 10;
    double vorher = vect[0];
    while(idx < DIM)
    {
        if(vect[idx] >= vorher) steigt++;
        if(vect[idx] <= vorher) faellt++;
        vorher = vect[idx++];
    }

    if(steigt == DIM - 1)
        ret = 11;
    else if(faellt == DIM - 1)
        ret = 12;
    return ret;      // Welchen Wert hat 'idx' in dieser Zeile?
}

void fill_random(void)
{  
}
```

- 2.1. Vervollständigen Sie die switch-case-Anweisung im Hauptprogramm main().
- 2.2. Schreiben Sie die Definition der Funktion fill\_random(). Diese Funktion soll mithilfe einer Schleife alle Elemente des Vektors „vect“ mit unterschiedlichen double-Zufallszahlen im Bereich von 1,5 ... 2,5 (beide Grenzen eingeschlossen!) füllen.
- 2.3. Welchen Wert hat die Variable „idx“, wenn das Programm in der mit /// markierten Zeile angekommen ist?  
idx =

- 2.4. Zeichnen Sie ein Struktogramm, das den genauen Ablauf der Funktion test\_monoton() zeigt.

### Aufgabe 3: (ca. 10 Punkte)

- 3.1. Das folgende C-Programm soll ermitteln, wie oft der Buchstabe 'x' in einer Zeichenkette mit einer Länge von max. 50 Zeichen vorkommt. Korrigieren Sie die fünf Fehler im Quelltext!

```
#include <stdio.h>
int count_x(char *str);

int main(void)
{
    int erg;
    char str1[50];
    printf("Text eingeben: ");
    scanf("%50s", str1);
    erg = count_x(str1);
    printf("%d kleine x gefunden!\n, erg");
    return 0;
}

// Wie oft kommt ein kleines x im Text vor?
int count_x(char *str)
{
    int count = 0;
    while (*str != '0')
    {
        if (*str == 'x') count++;
        ++str;

        return count;
    }
}
```

- 3.2. Die fünf selbst programmierten Funktionen bildschirm\_loeschen(), text\_ausgeben(), potenzieren(), vektor\_fuellen() und mitternachtsformel() werden folgendermaßen aufgerufen:

```
bildschirm_loeschen();

char str[] = "Viel Erfolg bei der Klausur!";
text_ausgeben(str);

double basis = 10.0, ergebnis;
ergebnis = potenzieren(basis, 0.5);

#define ANZAHL 100
float my_vekt[ANZAHL];
vektor_fuellen(my_vekt, ANZAHL);

double a = 1.0, b = 0.0, c = -2.0, x1, x2;
int anzahl_nullstellen;
anzahl_nullstellen = mitternachtsformel(a, b, c, &x1, &x2);
```

Wie sehen die Deklarationen (Prototypen) der 5 Funktionen aus (mehrere Varianten möglich)?

#### Aufgabe 4: (ca. 12 Punkte)

- 4.1. Wandeln Sie die folgende Dezimalzahl ins Dualsystem (Binärsystem) um:  $245_{DEZ} = ???_{BIN}$   
Notieren Sie alle Zwischenschritte Ihrer Berechnung!

$$\begin{array}{r} 245 \\ - 128 \\ \hline 117 \\ - 64 \\ \hline 53 \\ - 32 \\ \hline 21 \\ - 16 \\ \hline \end{array}$$

$$\begin{array}{ccccccccc} 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ \downarrow & \downarrow \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}$$

- 4.2. Wandeln Sie dieselbe Dezimalzahl auch ins Hexadezimalsystem um:  $245_{DEZ} = ???_{HEX}$

$$\begin{array}{rcl} 245 : 16 & = & 15 \quad R_{rest} \uparrow \\ 15 : 16 & = & 0 \quad R_{rest} \quad 15 = F \quad F5 \end{array}$$

- 4.3. In einer Speicherzelle im Arbeitsspeicher eines Computers ist die folgende 8-Bit-Binärzahl gespeichert: 0111 1111

Zu dieser Zahl wird 1 addiert.

- Wie lautet das Ergebnis der Addition als Dezimalzahl, wenn es sich bei der 8-Bit-Binärzahl um eine ganze Zahl ohne Vorzeichen handelt?

$$\begin{array}{r} 0111 \quad 1111 \\ + 0000 \quad 0001 \\ \hline 1000 \quad 0000 \end{array} \quad 127$$

- Wie lautet das Ergebnis der Addition als Dezimalzahl, wenn es sich bei der 8-Bit-Binärzahl um eine ganze Zahl mit Vorzeichen (also: 8-Bit-Zweierkomplementdarstellung) handelt?

$$127$$

- 4.4. In einem C-Programm ist eine Matrix wie folgt definiert:

```
double mat[10][10][2];
```

Wie viele Speicherzellen (mit jeweils 8 Bits pro Speicherzelle) werden benötigt, um diese Matrix im Arbeitsspeicher eines Computers abzuspeichern.

$$10 \cdot 10 \cdot 2 \cdot 8 \cdot 8 = 10.000 \cdot 8 \text{ Bit} = 10.000 \text{ Byte}$$



# Ingenieurinformatik

## C-Programmierung

Name	Vorname	Semestergruppe	Hörsaal

	Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5	Summe

**Beginn Ihres Bachelorstudiums:**

- vor WS13/14 (Kombiprüfung C + MATLAB) \*\*
- von WS13/14 bis WS15/16 \*\*
- von SS16 bis WS17/18 (Kombiprüfung C + MATLAB)
- ab SS18

**\*\* Die Prüfung ist nur dann gültig, wenn Sie die Zulassungsvoraussetzung erworben haben (erfolgreiche Teilnahme am Praktikum).**

**Aufgabensteller:** Dr. Reichl, Dr. Küpper und Kollegen

**Bearbeitungszeit:** 60 Minuten

**Hilfsmittel:** Taschenrechner nicht zugelassen,  
PC/Notebook nicht zugelassen,  
sonstige eigene Hilfsmittel sind erlaubt,  
Bearbeitung mit Bleistift ist erlaubt.

**\*\*\* Viel Erfolg! \*\*\***

### **Aufgabe 1: (ca. 25 Punkte)**

Schreiben Sie ein C-Programm, welches das bestimmte Integral der Funktion  $f(x) = (\sin x)^2$  auf dem Intervall  $[a; b]$  mit dem „Monte-Carlo-Verfahren“ numerisch ermittelt:

1. Zwei Zählervariablen  $z1$  und  $z2$  werden definiert und zunächst auf 0 gesetzt.
2. Der Anwender gibt die Integrationsgrenzen  $a$  und  $b$  ein.
3. Die folgenden Schritte 3.1. bis 3.3. werden in einer Schleife 1000-fach wiederholt. Auf den Aufruf der Funktion `rand()` zur Initialisierung des Zufallsgenerators kann verzichtet werden:
  - 3.1. Es wird ein zufälliger  $x$ -Wert (Datentyp „double“) im Bereich  $a \dots b$  erzeugt.
  - 3.2. Es wird ein zufälliger  $y$ -Wert (Datentyp „double“) im Bereich 0 ... 1 erzeugt.
  - 3.3. Falls  $f(x) < y$  ist, dann wird der Zähler  $z1$  inkrementiert, ansonsten wird  $z2$  inkrementiert.
4. Anschließend wird der Näherungswert  $f_n$  für das gesuchte Integral wie folgt ermittelt:

$$f_n = \frac{b - a}{z1 + z2} \cdot z2$$

5. Ihr Programm berechnet zusätzlich zum Näherungswert  $f_n$  auch den exakten Wert des Integrals. Dies ist leicht möglich, weil die Stammfunktion von  $f(x)$  bekannt ist:

$$F(x) = \frac{1}{2} \cdot (x - \sin x \cdot \cos x)$$

6. Auf dem Bildschirm werden drei Ergebnisse ausgegeben:
  - der numerisch berechnete Näherungswert  $f_n$ ,
  - der mithilfe der Stammfunktion berechnete exakte Wert,
  - die prozentuale Abweichung der beiden Werte (Ausbau des Prozentzeichens nicht vergessen!).

**Die Funktionswerte von  $f(x)$  bzw.  $F(x)$  werden in zwei separaten C-Funktionen berechnet, also nicht direkt im Hauptprogramm `main()`. An diese beiden Funktionen wird jeweils ein einzelner  $x$ -Wert übergeben, der daraus berechnete Wert  $f(x)$  bzw.  $F(x)$  wird anschließend ans Hauptprogramm `main()` zurückgegeben.**

Die Bildschirmausgabe Ihres Programms soll dem folgendem Bildschirmfoto entsprechen:

```
C:\Qt\Tools\QtCreator\bin\qtcreator_prj
a: 0.0
b: 6.283
Numerisch: 3.148
Exakt: 3.142
Abweichung: 0.197 %
```

*Eingabe der Intervallgrenzen a und b durch den Anwender*

*Ausgabe der Ergebnisse (Ausbau des Prozentzeichens nicht vergessen!)*

```

import random
import math

def f(x):
    return math.sin(x) ** 2

def F(x):
    return 0.5 * (x - math.sin(x) * math.cos(x))

def monte_carlo_integration(a, b, n_samples):
    z1 = 0
    z2 = 0
    for i in range(n_samples):
        x = random.uniform(a, b)
        y = random.uniform(0, 1)
        if f(x) < y:
            z1 += 1
        else:
            z2 += 1
    fn = (b - a) / (z1 + z2) * z2
    exact = F(b) - F(a)
    return fn, exact

if __name__ == "__main__":
    a = float(input("Bitte geben Sie die obere Intervallgrenze an: "))
    b = float(input("Bitte geben Sie die untere Intervallgrenze an: "))
    n_samples = 1000

    fn, exact = monte_carlo_integration(a, b, n_samples)

    print("Der numerisch berechnete Näherungswert:", fn)
    print("Der mithilfe der Stammfunktion berechnete exakte Wert:", exact)
    print(["Die prozentuale Abweichung: {:.2f}%".format(100 * abs(fn - exact) / exact)])

```

### **Aufgabe 2: (ca. 18 Punkte)**

Die drei globalen Matrizen m1, m2 und m3 sind wie folgt definiert. Gehen Sie davon aus, dass alle drei Matrizen **zunächst vollständig mit zufälligen Werten belegt** sind:

```
#define N 10
double m1[N][N], m2[N][N], m3[N][N];
```

2.1. Schreiben Sie eine Funktion fill1(), welche mithilfe von geeigneten for-Schleifen die Elemente von m1 wie folgt belegt. Verwenden Sie keine do-while-Schleifen und keine while-Schleifen:

1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0
11.0	12.0	13.0	14.0	15.0	16.0	17.0	18.0	19.0	20.0
21.0	22.0	23.0	24.0	25.0	26.0	27.0	28.0	29.0	30.0
31.0	32.0	33.0	34.0	35.0	36.0	37.0	38.0	39.0	40.0
41.0	42.0	43.0	44.0	45.0	46.0	47.0	48.0	49.0	50.0
51.0	52.0	53.0	54.0	55.0	56.0	57.0	58.0	59.0	60.0
61.0	62.0	63.0	64.0	65.0	66.0	67.0	68.0	69.0	70.0
71.0	72.0	73.0	74.0	75.0	76.0	77.0	78.0	79.0	80.0
81.0	82.0	83.0	84.0	85.0	86.0	87.0	88.0	89.0	90.0
91.0	92.0	93.0	94.0	95.0	96.0	97.0	98.0	99.0	100.0

```
void fill1(void)
{
```

```
}
```

2.2. Erstellen Sie ein Struktogramm, welches den genauen Ablauf der Funktion fill1() zeigt:

- 2.3. Schreiben Sie eine Funktion fill2(), welche mithilfe von geeigneten do-while-Schleifen die Elemente von m2 wie folgt belegt. Verwenden Sie keine while-Schleifen und keine for-Schleifen:

```
10.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  
 0.0 10.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  
 0.0  0.0 10.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  
 0.0  0.0  0.0 10.0  0.0  0.0  0.0  0.0  0.0  0.0  
 0.0  0.0  0.0  0.0 10.0  0.0  0.0  0.0  0.0  0.0  
 0.0  0.0  0.0  0.0  0.0 10.0  0.0  0.0  0.0  0.0  
 0.0  0.0  0.0  0.0  0.0  0.0 10.0  0.0  0.0  0.0  
 0.0  0.0  0.0  0.0  0.0  0.0  0.0 10.0  0.0  0.0  
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0 10.0  0.0  
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0 10.0
```

```
void fill2(void)  
{
```

```
}
```

- 2.4. Erstellen Sie ein Struktogramm, welches den genauen Ablauf der Funktion fill2() zeigt:

### Aufgabe 3: (ca. 7 Punkte)

Das folgende Programm sucht alle Primzahlen im Bereich von 2 bis 10000. Die Primzahlen werden auf dem Bildschirm ausgegeben, ebenso die Anzahl der gefundenen Primzahlen.

- Korrigieren Sie die fünf Fehler, die sich im Quelltext befinden!
- Wie oft wird die mit //1// markierte Zeile ausgeführt?
- Welchen Wert hat die Variable n in der mit //2// markierten Zeile?

```
#include <math.h>
int main(void)
{
    int n, t, teiler_gefunden, anzahl = 0;
    for(n = 2; n <= 10000; ++n)
    {
        teiler_gefunden = 0;      //1//
        for(t = 2; t < n && teiler_gefunden == 0; t += 1)
            if(n % t == 0) teiler_gefunden = 1;
        if(teiler_gefunden == 0) {
            printf("%16d", n); anzahl = anzahl++;
        }
    }
    printf("\n%d Primzahlen gefunden\n", &anzahl); //2//
    return 0;
}
```

Anzahl:

n =

### Aufgabe 4: (ca. 6 Punkte)

Das folgende Programm dient zur Verschlüsselung von Texten. In den mit //// markierten Zeilen werden nacheinander die Zeichenketten s1, s2, s3 an die Funktion crypto() übergeben. Fügen Sie die fehlenden Übergabeparameter zum Funktionsaufruf hinzu!

Wie sieht die Ausgabe des Programms auf dem Bildschirm aus?

```
#include <stdio.h>
#include <string.h>
void crypto(char *str);

int main()
{
    char s1[] = "Ingenieur", s2[] = "", s3[] = "x";

    crypto([REDACTED]); printf("*%s*\n", s1); //// Übergabe von s1
    crypto([REDACTED]); printf("*%s*\n", s2); //// Übergabe von s2
    crypto([REDACTED]); printf("*%s*\n", s3); //// Übergabe von s3

    return 0;
}

void crypto(char *str)
{
    char ch;
    int i, len = (int)strlen(str);
    for(i = 0; i < len / 2; ++i)
    {
        ch = str[i];
        str[i] = str[len - 1 - i];
        str[len - 1 - i] = ch;
    }
}
```

Ausgabe des Programms:

### Aufgabe 5: (ca. 11 Punkte)

Gegeben ist die folgende 8-stellige Binärzahl: **11100110**

- 5.1. Wandeln Sie diese (vorzeichenlose) Binärzahl in eine Hexadezimalzahl um.

$$\begin{array}{r} 230 \quad 1E6 \\ \text{Bin.} \quad 1110 \quad 0110 \\ \text{Dec} \quad 14 \quad 6 \\ \text{E} \quad 6 \end{array}$$

- 5.2. Bei der angegebenen Binärzahl handelt es sich um eine ganze Zahl mit Vorzeichen in 8-Bit-Zweierkomplementdarstellung. Welchen Wert hat die entsprechende Dezimalzahl?

<u>Zweier-Komplement</u>	$\begin{array}{r} 11100110 \\ + 00011001 \\ \hline 00011010 \end{array}$	<u>Einer-Komplement</u>	$\begin{array}{r} 11100110 \\ - 00011001 \\ \hline - 25 \end{array}$
--------------------------	--	-------------------------	--

- 5.3. Bei der angegebenen Binärzahl handelt es sich nun um eine ganze Zahl ohne Vorzeichen (also keine Zweierkomplementdarstellung). Welchen Wert hat die entsprechende Dezimalzahl?

$$230$$

- 5.4. Ein Mikrocontroller mit 8 Bit großen Registern addiert zur oben angegebenen Binärzahl die folgende 8-stellige Binärzahl hinzu: **01100000**

$$\begin{array}{r} 01100000 \\ + 11100110 \\ \hline 100000110 \end{array} \quad \text{Carry Flag gesetzt}$$

- 5.5. Ein Mikrocontroller mit 8 Bit großen Registern addiert zur oben angegebenen Binärzahl die folgende 8-stellige Binärzahl hinzu: **00000110**

Wie lautet der Inhalt des Ergebnisregisters? Ist das Carry-Flag im Mikrocontroller gesetzt, d. h. kommt es zu einem Übertrag an der höchstwertigen Stelle der Binärzahl?

$$\begin{array}{r} 00000110 \\ + 11100110 \\ \hline 11101100 \end{array} \quad \text{Carry Flag nicht gesetzt}$$



# Ingenieurinformatik

## C-Programmierung

Name	Vorname	Semestergruppe	Hörsaal

	Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5	Summe

### Bachelorstudiengang:

- Studienbeginn vor WS13/14 (Kombinationsprüfung) \*\***
- Studienbeginn ab WS13/14 bis WS15/16 \*\***
- Studienbeginn ab SS16 (Kombinationsprüfung)**
  
- Diplomstudiengang Fahrzeugtechnik \*\***

**\*\* Die Prüfung ist nur dann gültig, wenn Sie die Zulassungsvoraussetzung erworben haben (erfolgreiche Teilnahme am Praktikum).**

**Aufgabensteller: Dr. Reichl, Dr. Küpper und Kollegen**

**Bearbeitungszeit: 60 Minuten**

**Hilfsmittel:** Taschenrechner nicht zugelassen,  
PC/Notebook nicht zugelassen,  
sonstige eigene Hilfsmittel sind erlaubt,  
Bearbeitung mit Bleistift ist erlaubt.

**\*\*\* Viel Erfolg! \*\*\***

### Aufgabe 1: (ca. 22 Punkte)

Erstellen Sie ein C-Programm, welches für  $n$  Messwertpaare  $x_i, y_i$  (mit  $i = 1 \dots n$ ) die dazugehörige Ausgleichsgerade  $y = m \cdot x + b$  berechnet.

- Nach dem Programmstart wird der Anwender nach der Anzahl  $n$  der Messwertpaare gefragt. Die Anzahl muss zwischen 3 und 100 liegen, ansonsten wird die Eingabe wiederholt. Es darf aber davon ausgegangen werden, dass der Anwender nur Zahlen (im korrekten Format) eingibt.
- Nun wird der Anwender der Reihe nach zur Eingabe der einzelnen x- und y-Werte aufgefordert (Datentyp **double** verwenden; siehe Bildschirmfoto unten auf dieser Seite!).
- Anschließend werden der Mittelwert der x-Werte (im Folgenden als  $\bar{x}$  bezeichnet) und der Mittelwert der y-Werte (im Folgenden als  $\bar{y}$  bezeichnet) mit vier Nachkommastellen ausgegeben.
- Schließlich berechnet das Programm die Koeffizienten  $m$  und  $b$  der Ausgleichsgeraden und gibt beide Werte ebenfalls mit vier Nachkommastellen aus.
- Alle Eingabeaufforderungen und Ausgaben sollen so realisiert werden, wie es im Bildschirmfoto unten auf dieser Seite gezeigt ist.

Für den Koeffizienten  $m$  gilt:

$$m = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

Für den Koeffizienten  $b$  gilt:

$$b = \bar{y} - m \cdot \bar{x}$$

Die Bildschirmausgabe des C-Programms soll folgendermaßen aussehen:

```
C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe

Anzahl der Punkte (3...100): 110
Anzahl der Punkte (3...100): 1
Anzahl der Punkte (3...100): 4
x1 = 1.0
y1 = 0.5
x2 = 2.0
y2 = 2.0
x3 = 3.0
y3 = 1.5
x4 = 4.0
y4 = 2.5
Mittelwert der x-Werte = 2.5000
Mittelwert der y-Werte = 1.6250
Ausgleichsgerade y = m * x + b mit m = 0.5500, b = 0.2500
```

```

import math as m
    •••

n= int(input("Bitte geben Sie die Anzahl der Messwertpaare an (zwischen 3 und 100): "))

Liste = []
xi = 0
yi = 0
xi1 = 0
yi1 = 0
xi2 = 0
yi2 = 0
for i in range(n):
    x = float(input(f"x{i + 1} = "))
    y = float(input(f"y{i + 1} = "))
    print("")
    Liste.append([x, y])

for l in range(n):
    x = Liste[l][0]
    xi += x

for l in range(n):
    y = Liste[l][1]
    yi += y

x_mittel = xi/n
y_mittel = yi/n

for i in range(n):
    xi = Liste[i][0]
    xi2 = xi - x_mittel
    xi2 += xi2

    yi = Liste[i][1]
    yi2 = yi - y_mittel
    yi2 += yi2

m1 = (xi2 * yi2)/(xi2*xi2)
b = y_mittel-m1*x_mittel

print(f"Mittelwerte der x-Werte = {x_mittel:.4f}")
print(f"Mittelwerte der y-Werte = {y_mittel:.4f}")
print(f"Ausgleichsgerade y = m * x + b mit m = {m1:.4f}, b = {b:.4f}")

```

### **Aufgabe 2: (ca. 22 Punkte)**

Die globale Variable **data** ist wie folgt definiert: **double data[4][8][32];**

2.1. Schreiben Sie eine C-Funktion **fill1**, welche alle Elemente von **data** auf den Wert 1 (eins) setzt.  
Schreiben Sie Ihre Funktion unter Verwendung geeigneter **for-Schleifen**.

2.2. Erstellen Sie ein Struktogramm, welches den Ablauf der Funktion **fill1** zeigt.

**Lösung zu 2.1 (Quelltext)**

```
void fill1(void)
{
}
```

**Lösung zu 2.2 (Struktogramm)**

2.3. Schreiben Sie nun eine C-Funktion **fill2**, welche alle Elemente von **data** auf den Wert 2 (zwei) setzt.  
Schreiben Sie diese Funktion unter Verwendung geeigneter **do-while-Schleifen**.

2.4. Erstellen Sie auch ein Struktogramm, welches den Ablauf der Funktion **fill2** zeigt.

**Lösung zu 2.3 (Quelltext)**

```
void fill2(void)
{
}
```

**Lösung zu 2.4 (Struktogramm)**

### **Aufgabe 3: (ca. 10 Punkte)**

Schreiben Sie eine Funktion **woerter\_zählen**, welche die Anzahl der Wörter in einer Zeichenkette ermittelt und als Rückgabewert zurückgibt. Die einzelnen Wörter bestehen aus mindestens einem Buchstaben und sind durch ein (oder mehrere!) Leerzeichen voneinander getrennt.

Ihre Funktion soll auch dann ein korrektes Ergebnis liefern, wenn sich am Anfang der Zeichenkette Leerzeichen befinden oder wenn die Zeichenkette leer ist.

**Hinweis:** Zur Vereinfachung dürfen Sie davon ausgehen, dass sich in der Zeichenkette lediglich Buchstaben und Leerzeichen befinden – keine anderen Zeichen.

```
#include <stdio.h>
int woerter_zählen(char *s);

int main(void)
{
    printf("%d\n", woerter_zählen("a bb ccc"));      // Ausgabe: 3
    printf("%d\n", woerter_zählen("      "));          // Ausgabe: 0
    printf("%d\n", woerter_zählen(""));                // Ausgabe: 0
    printf("%d\n", woerter_zählen("  xx  yy  "));     // Ausgabe: 2
    return 0;
}

int woerter_zählen(char *s)
{
```

**Aufgabe 4: (ca. 7 Punkte)**

- 4.1. Das folgende C-Programm berechnet die Bewegung eines Feder-Masse-Schwingers mit dem Eulerverfahren. (Das zu lösende Differentialgleichungssystem besteht aus zwei Differentialgleichungen erster Ordnung.) Korrigieren Sie die fünf Fehler im C-Quelltext!

```
/* Schwingungs-DGL mit Eulerverfahren lösen */
#include <stdio.h>

#define DELTA_T 0.001; /* Zeitschritt in Sekunden */
#define SCHRITTE 10000 /* Anzahl der Zeitschritte */

void main(void)
{
    double y[1] = { 1.0, 0.0 }; /* Anfangswerte */
    double k = 1.0, m = 1.0; /* Federkonstante, Masse */
    double y_punkt[2], t = 0;

    int i;
    for(i = 0; i < SCHRITTE; ++i)
    {
        /* Ableitung berechnen */
        y_punkt[0] = y[1];
        y_punkt[1] = -k/m * y[0];

        /* Eulerverfahren */
        y[0] = y[0] + DELTA_T * y_punkt[0];
        y[1] = y[1] + DELTA_T * y_punkt[1];
        t = t + DELTA_T;

        /* Aktuelle Zeit und Auslenkung ausgeben */
        printf("%.4f ; %.4f \n, t, y[0]");
    }

    /* XXX Welchen Wert hat i an dieser Stelle? XXX */
    return 0;
}
```

- 4.2. Welchen Wert hat i nach dem Ende der for-Schleife, also in der mit /\* XXX \*/ markierten Zeile?

i =

- 4.3. Wie oft werden die Befehle innerhalb der for-Schleife durchlaufen?

Anzahl der Schleifen-Durchläufe =

**Aufgabe 5: (ca. 6 Punkte)**

- 5.1. Wandeln Sie die folgende Dezimalzahl in eine Dualzahl (Binärsystem) um. Geben Sie nicht nur das Endergebnis, sondern auch alle Zwischenschritte Ihrer Berechnung an!

$$199_{\text{DEZ}} = ??_{\text{BIN}}$$

- 5.2. Wandeln Sie dieselbe Zahl  $199_{\text{DEZ}}$  ins Hexadezimalsystem um!

$$199_{\text{DEZ}} = ??_{\text{HEX}}$$

(Platz für Notizen und Nebenrechnungen)