

# Ingenieurinformatik

## C-Programmierung

Name	Vorname	Matrikelnummer	Sem.-Gr.:	Hörsaal	Platz

Zulassung geprüft:

	Aufgabe 1	Aufgabe 2	Aufgabe 3	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.

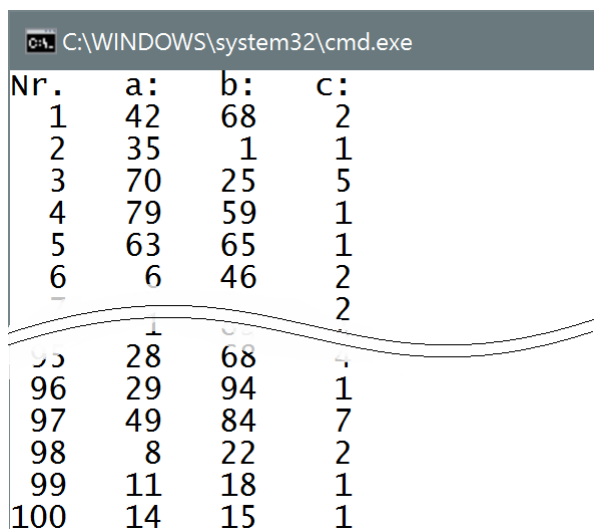
**\*\*\* Viel Erfolg! \*\*\***

### Aufgabe 1: (ca. 21 Punkte)

- 1.1. Programmieren Sie eine Funktion **int ggT(int a, int b)** zur Berechnung des größten gemeinsamen Teilers (ggT) von zwei Integer-Werten a und b. Zur Berechnung ist der „euklidische Algorithmus“ zu verwenden:

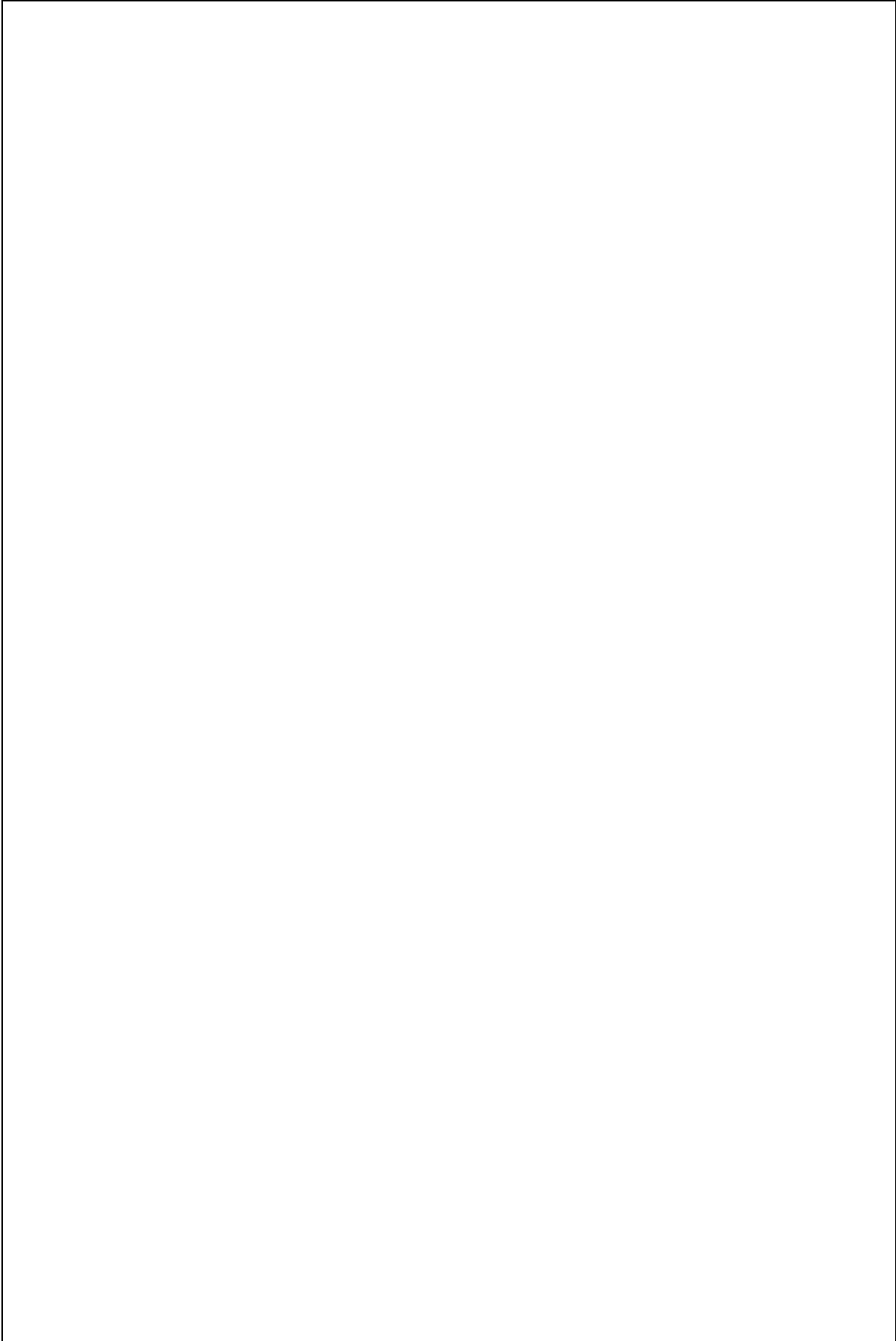
Solange b≠0 ist..
Hilfsvariable tmp = a modulo b
a = b
b = tmp
Der gesuchte ggT befindet sich nun in der Variablen a

- 1.2. Erstellen Sie ein Hauptprogramm **int main(void)**, welches die folgenden Aufgaben bearbeitet:
- Es werden drei Vektoren a[], b[] und c[] mit jeweils 100 Elementen vom Typ „int“ definiert.
  - Die beiden Vektoren a[] und b[] werden mit jeweils 100 ganzzahligen Zufallszahlen im Bereich 1...100 gefüllt.
  - Mithilfe der im Unterpunkt 1.1. implementierten Funktion wird der ggT von a[0] und b[0] berechnet und in c[0] gespeichert. Anschließend wird der ggT von a[1] und b[1] berechnet und in c[1] gespeichert. Danach wird der ggT von a[2] und b[2] berechnet und in c[2] gespeichert. Und so weiter, bis für alle Werte in den Vektoren a[] und b[] die größten gemeinsamen Teiler berechnet und im Vektor c[] gespeichert wurden.
  - Erst danach, also wenn alle drei Vektoren vollständig belegt worden sind, erfolgt die Ausgabe der Ergebnisse auf dem Bildschirm wie folgt:
  - Die in den drei Vektoren gespeicherten Werte werden tabellarisch (rechtsbündig!) untereinander ausgegeben. Die Ausgabe soll so aussehen, wie in der folgenden Abbildung gezeigt.
  - Denken Sie auch an die Ausgabe der Überschrift.



```
C:\WINDOWS\system32\cmd.exe
Nr.  a:  b:  c:
1   42  68  2
2   35  1   1
3   70  25  5
4   79  59  1
5   63  65  1
6    6  46  2
-   -   -   -
95  28  68  1
96  29  94  1
97  49  84  7
98   8  22  2
99  11  18  1
100 14  15  1
```

- 1.3. Ergänzen Sie ggf. den von Ihnen geschriebenen C-Quelltext auf der nebenstehenden Seite 3 so, dass sich insgesamt ein vollständiges C-Programm ergibt, welches ohne Fehler vom Compiler übersetzt werden kann. (Denken Sie an die notwendigen Include-Dateien, Funktionsdeklarationen usw.)



## Aufgabe 2: (ca. 24 Punkte)

Das folgende C-Programm multipliziert zwei 3x3-Matrizen miteinander. Die Funktion zur Ausgabe des Ergebnisses muss noch programmiert werden.

```
#include <stdio.h>
#define DIM 3

void print_ergebnis(void);
void multiplikation(void);

double a[DIM][DIM] = { { 2, 0, 0 },
                       { 0, 2, 0 },
                       { 0, 0, 2 } };

double b[DIM][DIM] = { { 1, 2, 3 },
                       { 4, 5, 6 },
                       { 7, 8, 9 } };

double c[DIM][DIM] = { { 9, 8, 7 },
                       { 6, 5, 4 },
                       { 3, 2, 1 } };

int main(void)
{
    multiplikation();
    print_ergebnis();
    return 0;
}

void multiplikation(void)
{
    int z, s, idx;
    for(z = 0; z < DIM; ++z)
    {
        for(s = 0; s < DIM; ++s)
        {
            b[z][s] = 0;
            for(idx = 0; idx < DIM; ++idx)
                b[z][s] += a[z][idx] * c[idx][s];
        }
    }
}

void print_ergebnis(void)
{

```

Aufg.  
2.6

2.1. Zeichnen Sie ein Struktogramm der Funktion **void multiplikation(void)**.

2.2. Wird das Berechnungsergebnis in der Matrix a[], in b[] oder in c[] abgespeichert?

2.3. Wie lautet das Ergebnis der Matrixmultiplikation, wenn die Matrizen vor der Multiplikation mit den oben im Programm angegebenen Werten belegt werden?

2.4. Wie oft werden die mit **/\*\*A\*\*/** und **/\*\*B\*\*/** markierten Zeilen ausgeführt?

Zeile A:

Zeile B:

2.5. Wieviel Speicher (in Bytes) wird von den Matrizen a[], b[] und c[] insgesamt im Arbeitsspeicher des Rechners belegt, wenn das C-Programm auf einem Windows-PC ausgeführt wird?

Speicherbedarf insgesamt:

2.6. Programmieren Sie mit zwei verschachtelten Schleifen die Funktion **void print\_ergebnis(void)** zur Ausgabe der Ergebnismatrix. Jede Zeile der Ergebnismatrix wird in einer eigenen Zeile auf dem Bildschirm dargestellt. Die einzelnen Elemente der Matrix sollen mit zwei Nachkommastellen und mit einer Feldbreite von 6 Zeichen auf dem Bildschirm ausgegeben werden.

### Aufgabe 3: (ca. 22 Punkte)

- 3.1. Das folgende C-Programm ermittelt die Anzahl der Großbuchstaben in einer ASCII-Zeichenkette. Ergänzen Sie die fehlenden Anweisungen! Umlaute müssen nicht berücksichtigt werden!

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

[ ]

int main(void)
{
    char my_text[] = "Das ist ein Test!";

    int anz = [ ] // Funktion "gross" aufrufen,
                // "my_text" übergeben

    printf("Anzahl der Grossbuchstaben: %d\n", anz);
    return 0;
}

int gross( [ ] )
{
    int cnt = 0, idx;

    for(idx = 0; [ ] ; idx++)
    {
        if( [ ] ) cnt++;
    }
    return cnt;
}
```

- 3.2. Das folgende C-Programm dient zur Umwandlung von positiven Binär- in Dezimalzahlen (also keine 2er-Komplementdarstellung). Korrigieren Sie die fünf Fehler im Quelltext.

```
#include <stdio.h>

#define LAENGE 8
int bin_to_dez(int *binaer_ziffern)

int main(void)
{
    int zahl1[LAENGE] = { 1, 1, 1, 1, 1, 1, 1, 0 };
    int zahl2[LAENGE] = { 0, 1, 1, 1, 1, 1, 1, 1 };

    printf("zahl1 = %d\n", bin_to_dez(zahl1));
    printf("zahl2 = %d\n", bin_to_dez(zahl2));
    return 0;
}

int bin_to_dez(int *binaer_ziffern)
{
    int i, ergebnis;

    for(i = 0; i < LAENGE; ++i)
        ergebnis += binaer_ziffern[LAENGE-i-1] * (int)pow(2.0, i);
}
}
```

3.3. Wie sieht die Bildschirmausgabe des (korrigierten) C-Programms aus Unterpunkt 3.2 aus?

3.4. Im Unterpunkt 3.2 wird die Länge (= Anzahl der Binärziffern) der umzuwandelnden Binärzahlen über die Konstante „LAENGE“ eingestellt. Die „LAENGE“ der Binärzahlen darf allerdings nicht beliebig groß werden, sonst funktioniert das Programm nicht mehr korrekt. Warum?

3.5. Welches ist der maximale Wert für „LAENGE“, der bei den Windows-Rechnern im Praktikumsraum noch ohne Probleme möglich ist?

3.6. Wie lautet die Bildschirmausgabe des folgenden Programms?

```
#include <stdio.h>

void plus1(int x);
void plus2(int *y);

int main(void)
{
    int x = 1, y = 1;
    plus1(x);
    plus2(&y);
    printf("x = %d, y = %d\n", x, y);
    return 0;
}

void plus1(int x)
{
    x += 1;
}

void plus2(int *y)
{
    *y += 1;
}
```

**Bildschirmausgabe:**

**(Platz für Notizen und Nebenrechnungen)**