

# KLAUSUR 1 Programmierpraktikum 2BK11 6.11.2007 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:  
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!
- Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mitaufgenommen werden.
- Name der Quelldatei nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Die entsprechenden Quelldateien müssen auf Diskette abgespeichert und abgegeben werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

## AUFGABEN

1) Mit den folgenden Näherungsformeln soll der Flächeninhalt und der Umfang eines Kreises berechnet werden:

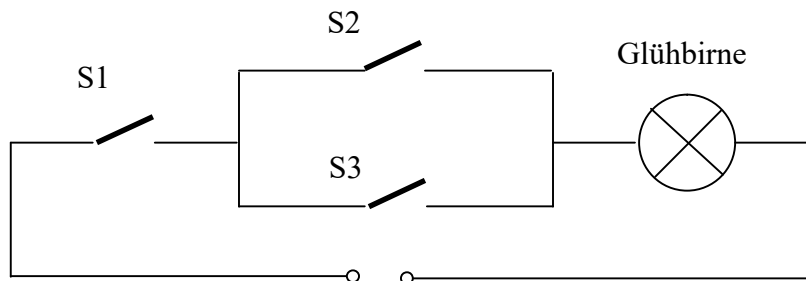
$$A = \frac{392}{126} \cdot r^2 \qquad U = 2 \cdot \frac{392}{126} \cdot r$$

Schreiben Sie ein C-Programm, das den Umfang und den Flächeninhalt eines Kreises berechnet (mit den obigen Formeln):

Der Radius soll über Tastatur eingegeben werden.

Auf dem Bildschirm soll dann der Umfang und der Flächeninhalt ausgegeben werden. (EVA-Prinzip beachten).

2) Eine Glühbirne wird mit 3 Schaltern wie folgt verbunden:



Jeder Glühbirne und jeder der 3 Schalter (S1, S2 und S3) hat entweder den Zustand 1 (Birne an bzw. Schalter an) oder 0 (Birne aus bzw. Schalter aus).

Schreiben Sie ein C-Programm, das den Zustand der Glühbirne berechnet:

Die Zustände der Schalter soll über Tastatur eingegeben werden.

Auf dem Bildschirm soll dann der Zustand der Glühbirne ausgegeben werden.

(EVA-Prinzip beachten). Kein if bzw, if - else verwenden

*Name, Vorname:*

Hilfsmittel:  
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

## AUFGABEN

- 1) Welche Eigenschaften hat ein Algorithmus ? (4 P)
- 2) Was ist die ANSI Norm ? (2 P)
- 3) Was ist ein Literal ? (2 P)
- 4) Was bedeutet Priorität in der Programmiersprache C ? (2 P)  
Geben Sie ein Beispiel.
- 5) Was bedeutet Assoziativität in der Programmiersprache C ? (2 P)  
Geben Sie ein Beispiel.
- 6) Kann die Ganzzahl 29999 in einer zwei Bytes grossen Integer-Zahl abgespeichert werden ?  
Begründen Sie! (4 P)
- 7) Das Volumen einer Kugel berechnet sich wie folgt: (10 P)  
$$V = \frac{4}{3} \cdot \pi \cdot r^3$$
  
Schreiben Sie das dazu passende C-Programm. Das Volumen muss in einer Variablen mit dem Datentyp float gespeichert werden. Die Konstante pi muss mit einer define-Direktive realisiert werden. Der Compiler darf **keine Warnungen** ausgeben.
- 8) (8 P)
  - a) Erscheint beim Kompilieren des folgenden syntaktisch korrekten Programmausschnitts eine Warnung ? Begründen Sie genau!
  - b) Welchen Wert hat f ?  
...  

```
float f;  
f = 3/5*(3.5 + 4 * 2.5);  
...
```

9) Was gibt der folgende Programmausschnitt auf dem Bildschirm aus ? Begründen Sie! (16P)

Bemerkung:

Werten Sie dazu jeweils die Ausdrücke (Bedingungen) in den if-Anweisungen aus.

```
-----  
int x=10;  
int a=0;  
int b=0;  
int c=0;  
if(x=2)  
    printf("Ausgabe1\n");  
if(a==b==c)  
    printf("Ausgabe2\n");  
if(123)  
    printf("Ausgabe3\n");  
if(5-3/4==5+3/4)  
    printf("Ausgabe4\n");  
if(x=(2&&8))  
    printf("Ausgabe5\n");  
    printf("Ausgabe6\n");  
if(x||b)  
    printf("Ausgabe7\n");  
if((x=2)&&8)  
    printf("Ausgabe8\n");  
-----
```

Lösung:

1) Ein Algorithmus ist ein Verfahren zur Lösung eines gegebenen Problems. Er ist eindeutig, endlich und schrittweise.

2) Die ANSI Norm ist eine Vorschrift, die festlegt, nach welchen Regeln ein C-Programm aufgebaut sein muss.

3) Literale sind Bezeichner mit einem festen Wert wie z.B. 9

4) Priorität gibt den Vorrang eines Operators in einem Ausdruck an.  
Beispiel: Punkt vor Strich.

5) Assoziativität gibt an, in welcher Reihenfolge (von links nach rechts oder von rechts nach links) Operatoren mit der gleichen Priorität abgearbeitet werden.

Beispiel: \* wird von links nach rechts abgearbeitet.

6) 2 Bytes kann  $2^{16} \approx 64000$  verschiedene Zustände speichern.

Damit können Ganzzahlen von ungefähr -32000 bis ungefähr 32000 abgespeichert werden.

Also kann 29999 in dieser zwei Bytes grossen Integer-Zahl gespeichert werden.

7)

```
#include "stdafx.h"
#include <stdio.h>
```

```
#define PI 3.14
```

```
int main(){
    double radius;
    float volumen;

    // Eingabeteil
    printf("Berechnung des Volumens einer Kugel\n");
    printf("Bitte geben sie den Radius ein\n");
    scanf("%lf",&radius);
    fflush(stdin);

    // Verarbeitungsteil
    volumen = (float) 4.0/3.0*PI*radius*radius*radius;

    //Ausgabeteil
    printf("Kugelvolumen= %f\n",volumen);

    return 0;
}
```

8)

a) Es wird zuerst  $3/5$  berechnet (Assoziativität von links nach rechts). Der Wert ist 0. 4 ist integer, 2.5 ist double. Also wird 4 in double 4.0 umgewandelt. Das Ergebnis 10.0 ist damit double. 3.5 ist double und damit ist  $3.5 * 10.0$  auch double. Dieser Wert wird mit integer 0 multipliziert, wobei deshalb vorher die integer 0 in double 0.0 umgewandelt wird und den Wert double 0.0 ergibt.

Da double in float abgespeichert wird, wird die double Zahl 0.0 in float umgewandelt. Dabei kann ein Datenverlust entstehen. Deshalb gibt der Compiler eine Warnung aus.

b) 0.0

9)

a) Ausgabe1: Der Wert des Ausdrucks (Zuweisungsausdrucks)  $x=2$  ist 2. Der Wert 2 wird als wahr interpretiert.

b) keine Ausgabe2: Der Wert des Ausdrucks  $b==c$  ist 1, also wahr, da b und c jeweils 0 ist und damit b gleich groß wie c ist. Der Wert des Ausdrucks  $a==(b==c)$ , also konkret  $a==1$ , ist 0, also falsch, da a gleich 0 ist. Der Wert 0 wird als falsch interpretiert.

c) Ausgabe3: 123 wird als wahr interpretiert.

d) Ausgabe4:  $5-3/4$  ist 5, da  $3/4$  Null ist.  $5+3/4$  ist 5, da  $3/4$  Null ist. Also ist  $5-3/4$  gleich  $5+3/4$ . Also hat der Ausdruck  $5-3/4==5+3/4$  den Wert 1 und wird als wahr interpretiert.

e) Ausgabe5: 2 und 8 wird jeweils als wahr interpretiert. Deshalb ist  $2 \ \&\& \ 8$  wahr und hat den Wert 1. Der Wert des Zuweisungsausdrucks  $x=(2\&\&8)$  ist deshalb 1 und wird als wahr interpretiert.

f) Ausgabe6: Nach `if(x=(2&&8))` steht keine Klammer {  
Deshalb besteht der if-Körper der dieser if-Anweisung nur aus einer Anweisung. Die dieser Anweisung folgende Anweisung wird damit auf jeden Fall ausgeführt.

g) Ausgabe7: Der Wert von x ist 2 und der Wert von b ist 0. Damit wird 2 als wahr und 0 als falsch interpretiert. Also hat der Ausdruck  $x \ || \ b$  den Wert 1 und ist damit wahr.

h) Ausgabe8: Der Wert des Ausdrucks (Zuweisungsausdrucks)  $x=2$  ist 2. Der Wert 2 wird als wahr interpretiert. 2 und 8 wird jeweils als wahr interpretiert. Deshalb ist  $2 \ \&\& \ 8$  wahr und hat den Wert 1. Der Wert des Zuweisungsausdrucks  $(x=2)\&\&8$  ist deshalb 1 und wird als wahr interpretiert.

# KLAUSUR 1 Programmierpraxis 2BKI1 Nachtermin 1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

## AUFGABEN

Schreiben Sie ein C-Programm, das den Ersatzwiderstand einer Parallelschaltung von 3 Widerständen berechnet (Datentyp jeweils float).

Wurde ein Widerstandwert kleiner oder gleich Null eingegeben, muß das Programm eine entsprechende Meldung auf den Bildschirm bringen (im Ausgabeteil !).

Dies muß mit dem EVA-Prinzip realisiert werden.

Bemerkung:

$$\frac{1}{R_G} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}$$

# KLAUSUR 1 Programmiertheorie 2BKI1 Nachtermin 1 Zeit: 30 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

## AUFGABEN

1) Es soll der Ersatzwiderstand einer Parallelschaltung von 3 Widerständen berechnet werden. Wurde ein Widerstandwert kleiner oder gleich Null eingegeben, muß das Programm sofort beendet werden und eine entsprechende Meldung auf den Bildschirm ausgegeben werden. (insbesondere darf dann nicht mehr ein weiterer Widerstand eingegeben und der Ersatzwiderstand berechnet werden).

Dies muß mit dem EVA-Prinzip realisiert werden.

Außerdem muß in der Ausgabe noch angegeben werden, welcher Widerstandswert (z.B. 1. Widerstandwert) falsch eingegeben wurde.


Erstellen Sie das dazugehörige **Struktogramm**.

Bemerkungen:

a) 
$$\frac{1}{R_G} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}$$

b) Es widerspricht nicht dem EVA-Prinzip, wenn z.B. im Eingabeteil in bestimmten Variablen bestimmte Zustände abgespeichert werden (z.B: ob der 3. Widerstandswert kleiner oder gleich 0 ist) und diese dann im Ausgabeteil abgeprüft werden.

**Widerstandsberechnung**

Eingabe r1				
<div>W<div>r1 &lt;= 0</div>F</div>				
status = -1	Eingabe r2			
	<div>W<div>r2 &lt;= 0</div>F</div>			
	status = -2	Einagbe r3		
		<div>W<div>r3 &lt;= 0</div>F</div>		
		status = -3		status=0
		status==0		
W			F	
rges = 1(1/r1+1/r2+1/r3)				
<div>W<div>status == -1</div>F</div>				
Ausgabe (1.Widerstand <=0)	<div>W<div>status == -2</div>F</div>			
	Ausgabe (2.Widerstand <=0)	<div>W<div>status == -3</div>F</div>		
		Ausgabe (3.Widerstand <=0)	Ausgabe(rges)	



*Name, Vorname:*

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

## AUFGABEN

1)

Definition:

Die Quersumme einer ganzen Zahl ist die Summe seiner Ziffern.

Beispiele:

Die Quersumme von 367 ist:  $3 + 6 + 7 = 16$

Die Quersumme von 1996 ist:  $1 + 9 + 9 + 6 = 25$

Die Quersumme von -13 ist: 4 (Quersumme ist immer  $\geq 0$ )

a) Erstellen Sie ein **Struktogramm**, das von einer ganzen Zahl die Quersumme berechnet.

Beachten Sie:

Die Quersumme ist immer  $\geq 0$

b) Machen Sie dazu verschiedene Tests (dokumentierte Protokolle mit Testdaten).

Geben Sie jeweils an, ob die Tests erfolgreich waren oder nicht.

## KLAUSUR 2 Programmiertheorie 2BKI1 Nachtermin 1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

Keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

### AUFGABEN

1) 20P

Erstellen Sie ein Struktogramm eines Programms, das die Fakultät einer ganzzahligen Zahl ( $\geq 1$ ) berechnet.

$$n! = n * (n-1) * (n-2) * \dots * 1$$

Beispiele:  $1! = 1$        $2! = 2 * 1$        $3! = 3 * 2 * 1$        $4! = 4 * 3 * 2 * 1$

Sie geben über Tastatur eine ganzzahlige Zahl ( $\geq 1$ ) ein. Von dieser Zahl soll die Fakultät berechnet werden und das Ergebnis auf dem Bildschirm ausgegeben werden.

Beispiel:

Sie geben z.B. die Zahl 4 ein. Auf dem Bildschirm wird dann ausgegeben:

$$4! = 24$$

2) 30P

Eine natürliche Zahl  $n$  ( $n \geq 2$ ) heißt Primzahl, wenn sie nur durch 1 und sich selbst teilbar ist.

Beispiele für Primzahlen: 2, 3, 5, 7, 11, 13, ....

Erstellen Sie ein Struktogramm eines Programms, das bestimmt, ob eine ganze Zahl  $n$  ( $n \geq 2$ ) eine Primzahl ist: Sie geben über Tastatur eine ganze Zahl  $n$  ( $n \geq 2$ ) ein. Auf dem Bildschirm soll dann ausgegeben werden, ob diese Zahl eine Primzahl ist oder nicht.

## KLAUSUR 2 Programmierpraxis 2BKI1 Nachtermin 1 Zeit: 90 Minuten

Name, Vorname:

Hilfsmittel:

Keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

### AUFGABEN

1) Was gibt der folgende Programmausschnitt auf dem Bildschirm aus ? Begründen Sie!

Bemerkung: (8P)

Werten Sie dazu jeweils die Ausdrücke (Bedingungen) in den if-Anweisungen aus.

```
int x=10;
int a=0;
int b=0;
int c=0;
if(x=2)
    printf("Ausgabe1\n");
if(a==b==c)
    printf("Ausgabe2\n");
if(123)
    printf("Ausgabe3\n");
if(5-3/4==5+3/4)
    printf("Ausgabe4\n");
if(x=(2&&8))
    printf("Ausgabe5\n");
    printf("Ausgabe6\n");
if(x||b)
    printf("Ausgabe7\n");
if((x=2)&&8)
    printf("Ausgabe8\n");
```

2) Ersetzen Sie folgenden syntaktisch korrekten Programmausschnitt durch eine verschachtelte ein- oder zweiseitige Verzweigung:

(5P)

Programmausschnitt:

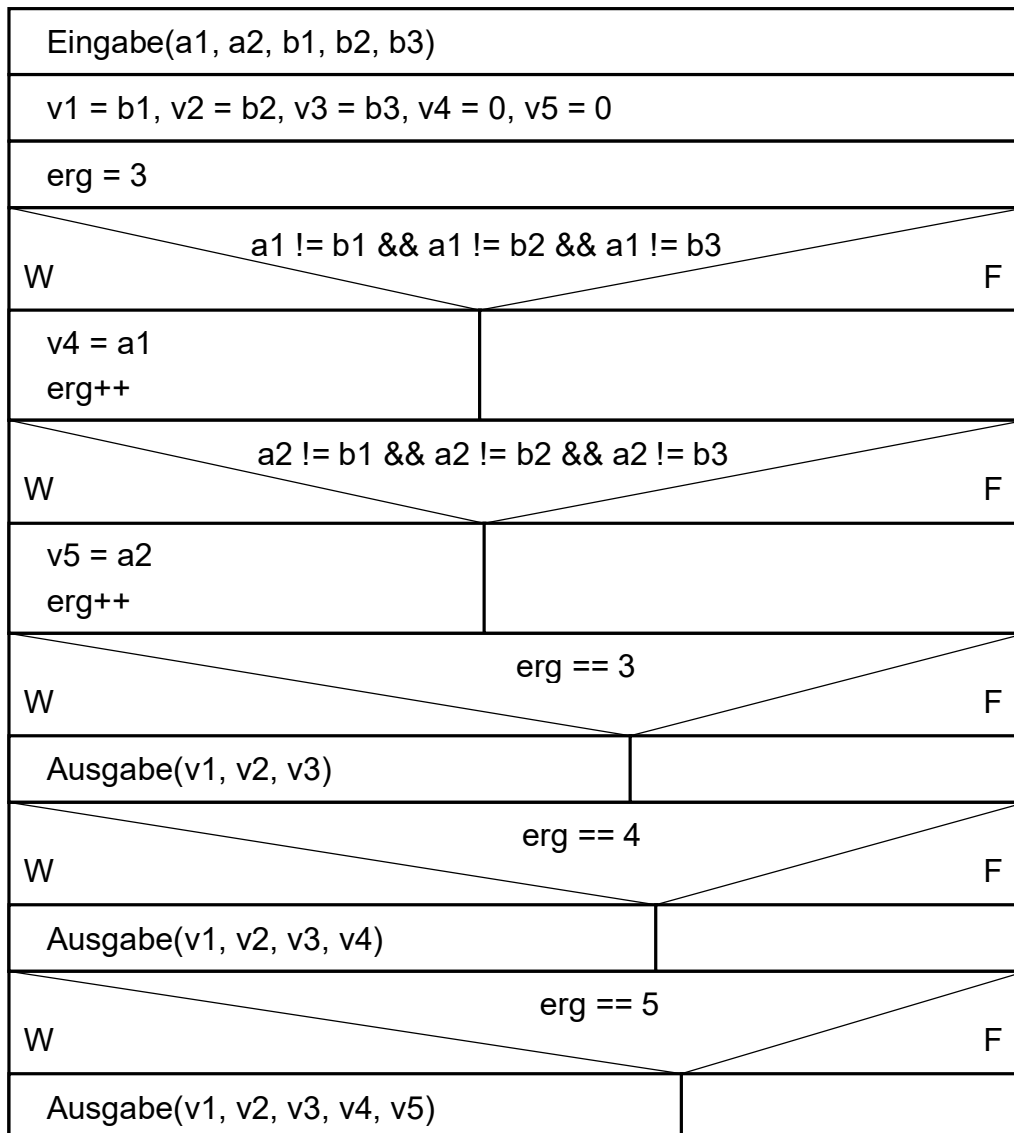
```
...
if(a==b && b==c)
    x=1;
else
    x=2;
```

3) Warum erscheint beim Kompilieren des folgenden syntaktisch korrekten Programmausschnitts eine Warnung ? Begründen Sie!

(5P)

```
...
float f;
f = 3.5 + 4 * 2.5;
```

- 4) Der durch das folgende Struktogramm beschriebene Algorithmus soll die Vereinigungsmenge der Zahlenmengen A und B mit  $A = \{a1, a2\}$  und der  $B = \{b1, b2, b3\}$  berechnen. (10P)
- Begründen Sie, ob der Algorithmus korrekt ist. Falls er nicht korrekt ist, muß dazu ein konkretes Gegenbeispiel angegeben werden (mit konkreten Werten für a1, a2, b1, b2, b3)



- 5) In den folgenden Programmausschnitten gibt der Anwender für i einen Wert ein. (5P)
- a) Für welche Werte von i bringen die Programmausschnitte die gleiche Anzahl von Meldungen auf den Bildschirm ?
- b) Für welche Werte von i bringen die Programmausschnitte nicht die gleiche Anzahl von Meldungen auf den Bildschirm ?

```
scanf("%d",&i);
while (i<=3){
    i = i+1;
    printf("Hallo Welt\n");
}
```

```
scanf("%d",&i);
do{
    i = i+1;
    printf("Hallo Welt\n");
} while(i<=3);
```

6) Gegeben ist der folgende syntaktisch korrekte Programmteil:

(4P)

```
i=10;
for(i=0;i<20;i=i+1){
    printf("Hallo Welt\n");
}
```

Wie oft gibt dieser Programmteil die Meldung "Hallo Welt" auf dem Bildschirm aus ?  
Begründen Sie !

7) Gegeben ist der folgende syntaktisch korrekte Programmteil (z1, z2 sind ganze Zahlen):

(8P)

```
scanf("%d",&z1);
scanf("%d",&z2);
sum = 0;
i = z1-1;
while(i<z2){
    i = i+1;
    sum = sum+i;
}
```

Was berechnet der Programmteil (Bedingungen für z1 und z2 angeben, wann was geschieht) ?

8) Gegeben ist der folgende Programmteil:

(5P)

```
sum = 0;
i = 1;

while(1==1){
    i = i+1;
    sum = sum +i;
-->
}
```

a)

Tragen Sie die Werte der Variablen (an der Stelle -->) i und sum bei den ersten drei Schleifendurchgängen in die Tabelle ein (ohne Berücksichtigung der Schleifenbedingung )

i					
sum					

b)

Verändern Sie den Programmausschnitt an genau 2 Stellen so, daß der Wert der Variablen sum nach Verlassen der Schleife gleich der Summe  $5 + 6 + 7 + \dots + 13 + 14 + 15$  ist.

## Lösungen:

1)a) Ausgabe1: Der Wert des Ausdrucks (Zuweisungsausdrucks)  $x=2$  ist 2. Der Wert 2 wird als wahr interpretiert.

b) keine Ausgabe2: Der Wert des Ausdrucks  $b==c$  ist 1, also wahr, da b und c jeweils 0 ist und damit b gleich groß wie c ist. Der Wert des Ausdrucks  $a==(b==c)$ , also konkret  $a==1$ , ist 0, also falsch, da a gleich 0 ist. Der Wert 0 wird als falsch interpretiert.

c) Ausgabe3: 123 wird als wahr interpretiert.

d) Ausgabe4:  $5-3/4$  ist 5, da  $3/4$  Null ist.  $5+3/4$  ist 5, da  $3/4$  Null ist. Also ist  $5-3/4$  gleich  $5+3/4$ . Also hat der Ausdruck  $5-3/4==5+3/4$  den Wert 1 und wird als wahr interpretiert.

e) Ausgabe5: 2 und 8 wird jeweils als wahr interpretiert. Deshalb ist  $2 \ \&\& \ 8$  wahr und hat den Wert 1. Der Wert des Zuweisungsausdrucks  $x=(2\&\&8)$  ist deshalb 1 und wird als wahr interpretiert.

f) Ausgabe6: Nach `if(x=(2&&8))` steht keine Klammer {

Deshalb besteht der if-Körper der dieser if-Anweisung nur aus einer Anweisung. Die dieser Anweisung folgende Anweisung wird damit auf jeden Fall ausgeführt.

g) Ausgabe7: Der Wert von x ist 2 und der Wert von b ist 0. Damit wird 2 als wahr und 0 als falsch interpretiert. Also hat der Ausdruck  $x \ || \ b$  den Wert 1 und ist damit wahr.

h) Ausgabe8: Der Wert des Ausdrucks (Zuweisungsausdrucks)  $x=2$  ist 2. Der Wert 2 wird als wahr interpretiert. 2 und 8 wird jeweils als wahr interpretiert. Deshalb ist  $2 \ \&\& \ 8$  wahr und hat den Wert 1. Der Wert des Zuweisungsausdrucks  $(x=2)\&\&8$  ist deshalb 1 und wird als wahr interpretiert.

2)

```
if (a==b)
    if (b==c)
        x=1;
    else
        x=2;
else
    x=2;
```

3) 4 ist integer, 2.5 ist double. Also wird 4 in double umgewandelt. Das Ergebnis 10 ist damit double. 3.5 ist double und damit ist  $3.5 * 10$  auch double. Da double in float abgespeichert wird, wird die double Zahl  $3.5 * 10$  in float umgewandelt. Dabei kann ein Datenverlust entstehen. Deshalb gibt der Compiler eine Warnung aus.

4)

```
a1 = 3,
a2 = 100,
b1 = 3,
b2 = 4,
b3 = 5
```

5)

$i \leq 3$ : gleiche Anzahl von Meldungen

$i > 3$  : verschiedene Anzahl von Meldungen

6)

Ein Mal, weil

```
printf("Hallo Welt\n");
```

nicht zum Körper der for-Anweisung (Semikolon beachten !) gehört.

7)

a)

Fall  $z1 \leq z2$ : Die Summe zwischen  $z1$  und  $z2$  (je einschließlich) wird berechnet

$$\text{Summe} = z1 + (z1+1) + (z1+2) + (z1+3) + \dots + z2$$

Beispiel:  $z1 = 3, z2 = 7 \implies \text{Summe} = 3 + 4 + 5 + 6 + 7$

Fall  $z1 > z2$ : Die Summe ist 0.

8)

a)

i	2	3	4		
sum	2	2 + 3	2 + 3 + 4		

b)

```
sum = 0;
```

```
i = 4;
```

```
while(i < 15) {
```

```
    i = i + 1;
```

```
    sum = sum + i;
```

```
}
```

*Name, Vorname:*Hilfsmittel:  
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

**AUFGABEN**

1) Was ist ein Pointer ?

2P

2)

10P

```
...  
float e = 3.1;  
float f = 2.7;  
float *p;  
p = &e;  
*p = f + *p;
```

	Adresse	Wert
e	01350	
f	02460	
p	03570	

Welchen Wert hat e und p nachdem alle die obigen Anweisungen ausgeführt wurden?  
Tragen Sie die Werte in die Tabelle ein.

3) In einem C-Programm wurde folgende Variable deklariert:

6P

```
char w[4];
```

In dem Programm stehen genau die folgenden Anweisungen:

```
w[0] = 'w';  
w[1] = 'i';  
w[2] = 'r';
```

Es soll ein Programm(ausschnitt) geschrieben werden, in dem der Inhalt der Variablen w (also die Zeichenkette "wir") ausgegeben werden soll.

- a) Realisieren Sie dies durch die Verwendung der Formatierung "%s" in printf
- b) Realisieren Sie dies durch die Verwendung der Formatierung "%c" in printf

4) Wieviel Speicherplatz (in Bytes) benötigen die Variablen v1, v2, v3 des folgenden Programmausschnitts ?

3P

```
int v1[10];  
int v2[10][10];  
int v3[10][10][10];
```



5)

14P

Ein Text ist in einem zweidimensionalen Feld `ff` (Quadrat) abgespeichert. Um ihn vor neugierigen Blicken zu schützen wird er so codiert, dass die 1. Zeile mit der 1. Spalte, die 2. Zeile mit der 2. Spalte, usw. vertauscht werden.

Realisieren Sie das mit Hilfe eines Struktogramms.

Zeilenanzahl: `LEN`

Spaltenanzahl: `LEN`

(Als Zeilenanzahl und Spaltenanzahl keine feste Zahl wählen!!)

Tipp: Überlegen Sie sich, wo das Element `ff[i][j]` nach seiner Vertauschung steht!

6)

5P

Gegeben ist ein zweidimensionalen Feld `ff` mit der Zeilenanzahl `ANZZ` und der Spaltenanzahl `ANZS`.

Die 1. Zeile dieses zweidimensionalen Feld `ff` wird in das eindimensionale Feld `f` (am Anfang des Feldes `f` beginnend) kopiert.

Gleich dahinter wird die 2. Zeile des zweidimensionalen Feldes `ff` angefügt.

Gleich dahinter wird die 3. Zeile des zweidimensionalen Feldes `ff` angefügt, usw.

a) Welche Feldlänge hat `f` ?

b) Das zweidimensionale Feld `ff` wird zerstört.

Durch welchen Zugriff kann man jetzt im eindimensionalen Feld `f` auf ein Element des Feldes `ff` zugreifen, das vor der Zerstörung an der Stelle `ff[i][j]` abgespeichert wurde ?

Bemerkungen:

In den obigen Aufgaben ist unter *i-ter Zeile* bzw. *i-ter Spalte* natürlich **programmtechnisch** die *i-1-te Zeile* bzw. *i-1-te Spalte* gemeint.

Das heißt zum Beispiel, dass mit 1. Zeile bzw. 1. Spalte programmtechnisch die 0. Zeile bzw. 0. Spalte gemeint ist

## Lösungen

1) Ein Pointer ist eine Variable, deren Wert die Adresse eines Speicherplatzes (z.B. einer Variable) ist.

2)

```
...  
float e = 3.1;  
float f = 2.7;  
float *p;  
p = &e;  
*p = f + *p;
```

	Adresse	Wert
e	01350	5.8
f	02460	2.7
p	03570	01350

3) a)

```
w[3] = '\\0';  
printf("%s", w);
```

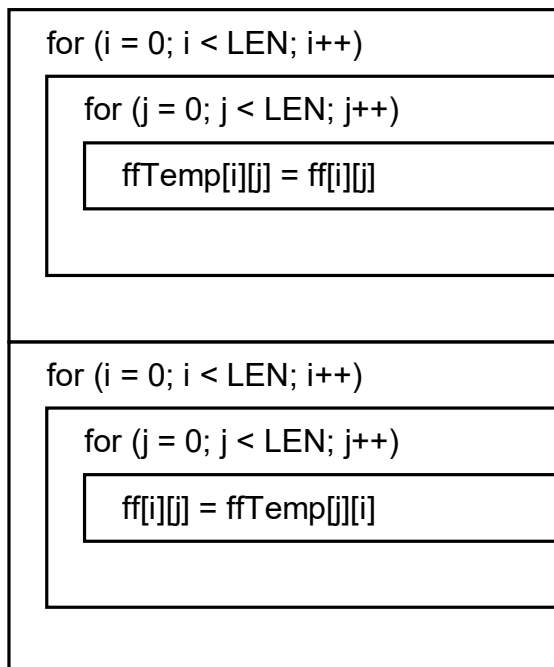
b)

```
for(i=0; i<3; i++)  
    printf("%c", w[i]);
```

4)

Speicherplatzverbrauch von v1 = 10 \* Speicherbedarf(int);  
Speicherplatzverbrauch von v2 = 100 \* Speicherbedarf(int);  
Speicherplatzverbrauch von v3 = 1000 \* Speicherbedarf(int);

5)



6)

a) (2P)

Feldlänge  $f = \text{ANZZ} * \text{ANZS}$

b) (3P)

$\text{ff}[i * \text{ANZS} + j] = \text{ff}[i][j]$

**KLAUSUR 3 Programmierpraktikum 2BK11 2.4.2008 Zeit: 45 Minuten**  
**Gruppe A Name, Vorname:**

Hilfsmittel:  
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!
- Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mitaufgenommen werden.
- Name der Quelldatei nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Die entsprechenden Quelldateien müssen auf Diskette abgespeichert und abgegeben werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

## AUFGABEN

1)

Schreiben Sie ein Programm, das die Potenz  $a^n$  einer reellen Zahl  $a$  bestimmt.

$n$  muß eine **ganze** Zahl sein.

Überlegen Sie sich welche(n) Parameter die Funktion hoch haben muss?

Im Hauptprogramm muss der Anwender eine ganze Zahl  $\geq 0$  eingeben können.

Das Programm berechnet dann (mit Hilfe der Funktion hoch(...)) das Ergebnis und gibt es auf dem Bildschirm aus.

Definition der Potenz:

$$a^n = \underbrace{a \cdot a \cdot \dots \cdot a}_{n - \text{mal}}, \text{ wenn } n > 0$$

$$a^n = 1, \text{ wenn } n = 0$$

$$a^{-n} = \frac{1}{a^n}, \text{ wenn } n < 0$$

Beispiele:

$$2^4 = 2 \cdot 2 \cdot 2 \cdot 2 = 16$$

$$5^0 = 1$$

$$3^{-2} = \frac{1}{3^2} = \frac{1}{9}$$

# KLAUSUR 3 Programmierpraktikum 2BK11 2.4.2008 Zeit: 45 Minuten

## Gruppe B Name, Vorname:

Hilfsmittel:  
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!
- Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mitaufgenommen werden.
- Name der Quelldatei nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Die entsprechenden Quelldateien müssen auf Diskette abgespeichert und abgegeben werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

## AUFGABEN

1)

Schreiben Sie ein Programm, das eine Multiplikationstabelle erstellt:

	1	2	3	4
1	1	2	3	4
2	2	4	6	8
3	3	6	9	12
4	4	8	12	16
5	5	10	15	20

Die Berechnung einer Zahl der Multiplikationstabelle wird mit der Funktion `berechneZahl(...)` realisiert.

Überlegen Sie sich welche(n) Parameter die Funktion `berechneZahl(...)` haben muss?

Das Programm berechnet dann (mit Hilfe der Funktion `berechneZahl(...)`) das Ergebnis und trägt es in die Tabelle ein.

Danach wird die gesamte Tabelle auf dem Bildschirm ausgegeben, so dass genau die oben angegebene Tabelle auf dem Bildschirm (ohne das Tabellengitter) erscheint.

Bemerkung:

Realisieren Sie das durch ein zweidimensionales Feld `ff`.

Zeilenanzahl: `ANZZEILEN`, Spaltenanzahl: `ANZSPALTEN`

Als Zeilenanzahl und Spaltenanzahl keine feste Zahl wählen, sondern wie üblich mit Konstanten arbeiten.

# KLAUSUR 3 Programmierpraxis 2BKI1 Nachtermin 1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

## AUFGABEN

1) Eine natürliche Zahl  $n$  ( $n \geq 2$ ) heißt Primzahl, wenn sie nur durch 1 und sich selbst teilbar ist. Beispiele für Primzahlen: 2, 3, 5, 7, 11, 13, ....

Schreiben Sie ein Programm, das folgendes macht:

a) In dem Programm muss sich eine Funktion `istPrimzahl(...)` befinden, die von einer ganzen Zahl  $\geq 2$  feststellt, ob diese eine Primzahl ist.

b) In dem Hauptprogramm sollen mit Hilfe der Funktion `istPrimzahl(...)` die ersten 100 Primzahlen ausgegeben werden.

# KLAUSUR 3 Programmiertheorie 2BKI1 Nachtermin 1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

## AUFGABEN

1) Wandeln Sie die folgende For-Anweisung in eine Do-While-Anweisung um:

```
for (A1; B; A3) {  
    A;  
}
```

2)

a)

Schreiben Sie eine Funktion

fak(...)

die die Fakultät einer ganzzahligen Zahl ( $\geq 1$ ) berechnet.

$n! = n * (n-1) * (n-2) * \dots * 1$

Beispiele:  $1! = 1$        $2! = 2*1$        $3! = 3*2*1$        $4! = 4*3*2*1$

b)

Im Hauptprogramm soll der Anwender über Tastatur eine ganzzahlige Zahl ( $\geq 1$ ) eingeben.

Von dieser Zahl soll mit Hilfe der Funktion fak(...) die Fakultät berechnet werden und das Ergebnis auf dem Bildschirm ausgegeben werden.

Beispiel:

Sie geben z.B. die Zahl 4 ein. Auf dem Bildschirm wird dann ausgegeben:

$n! = 24$

## Lösungen

5)

```
for (A1; B; A3)
  A;
```

$\Leftrightarrow$

```
A1;
while (B) {
  A;
  A3
}
```

$\Leftrightarrow$

```
A1;
if (B) {
  do {
    A;
    A3
  } while (B)
}
```



**KLAUSUR 4 Programmierpraktikum 2BK11 18.6.2008 Zeit: 45 Minuten**  
**Gruppe A Name, Vorname:**

Hilfsmittel:  
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!
- Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mitaufgenommen werden.
- Name der Quelldatei nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Die entsprechenden Quelldateien müssen auf Diskette abgespeichert und abgegeben werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

## AUFGABEN

1) 10P

a) Erstellen Sie (mit Hilfe eines C-Programms) die Datei "c:\bki1.txt", die die folgenden 4 Zeichen in diese Datei schreibt:  
"aBBa".

b) 40P

Erstellen Sie ein C-Programm, das in einer beliebigen Datei, die der Anwender über Tastatur eingibt, überall das in der Datei vorkommende Zeichen 'a' durch das Zeichen 'A' ersetzt.

**KLAUSUR 4 Programmierpraktikum 2BK11 18.6.2008 Zeit: 45 Minuten**  
**Gruppe B** *Name, Vorname:*

Hilfsmittel:  
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!
- Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mitaufgenommen werden.
- Name der Quelldatei nach folgendem Schema vergeben: Gruppe\_Rechnummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Die entsprechenden Quelldateien müssen auf Diskette abgespeichert und abgegeben werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

## AUFGABEN

1) 10P

a) Erstellen Sie (mit Hilfe eines C-Programms) die Datei "c:\e3fi.txt", die die folgenden Zeichen in diese Datei schreibt:  
"ÄltereändernAlles".

b) 40P

Erstellen Sie ein C-Programm, das in einer beliebigen Datei, die der Anwender über Tastatur eingibt, folgendes macht:

Ersetzen Sie die Umlaute Ä und ä dieser Datei durch "Ae" bzw. "ae" und speichern Sie den konvertierten Text in der Datei "C:\test2.txt" ab.

# KLAUSUR 4 Programmierpraxis 2BKI1 Nachtermin 1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

## AUFGABEN

50 P

Schreiben Sie **ein** fehlerfreies C-Programm, das folgendes macht:

Um die Datei "c:\geheim.txt" zu verschlüsseln, soll der Dateiinhalt in umgekehrter Reihenfolge angeordnet werden.

Beispiel:

Dateiinhalt vor der Verschlüsselung

i	n	f	o	r	m	a	t	i	k
---	---	---	---	---	---	---	---	---	---

Dateiinhalt nach der Verschlüsselung

k	i	t	a	m	r	o	f	n	i
---	---	---	---	---	---	---	---	---	---

**Wichtige Bemerkung:**

1) Das Programm soll so geschrieben werden, dass es nicht nur für z.B. eine Datei der Größe 10 Bytes funktioniert, sondern für eine beliebig große Dateien "c:\geheim.txt".

2) Falls nötig, dürfen natürlich auch (zusätzliche) temporäre Dateien erstellt werden

# KLAUSUR 4 Programmiertheorie 2BKI1 24.6.2008 Zeit: 60 Minuten

Name, Vorname:

Hilfsmittel:

Keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

## AUFGABEN

1)

12P

Betrachten Sie den folgenden Programmausschnitt:

```
...
int main() {
    int zahl = 4;
    int quad = 8;
    —————> [1]
    quadrat(zahl, &quad);
    —————> [2]
    printf("%d hoch 2 = %d\n", zahl, quad);
    return 0;
}

void quadrat(int z, int *zq) {
    *zq = z * z;
}
```

Durch die Deklaration der Variablen zahl und quad werden die folgenden Zellen

	Adresse	Inhalt
zahl	08151	?
quad	04711	?

im Arbeitsspeicher reserviert.

- Welchen Wert hat der Inhalt der Variablen zahl und quad an der Stelle [1] im Programm ?
- Welchen Wert hat z und zq beim Aufruf von quadrat(zahl, &quad) ?
- Was bewirkt die Anweisung \*zq = z \* z an welcher Adresse im Arbeitsspeicher ?  
(konkreten Wert der Adresse und deren Inhalt angeben!)
- Welchen Wert hat der Inhalt der Variablen zahl und quad an der Stelle [2] im Programm ?

2)

13P

Sie wollen ein Programm schreiben, das abhängig von der Eingabe entweder die Summe oder die Differenz oder das Produkt oder den Quotienten zweier Zahlen liefert.

Sie wollen dazu die Funktion `tr` (wie Taschenrechner) benutzen, die Sie aber wegen Arbeitsüberlastung von einem "Programmierknecht" implementieren (programmieren) lassen. Entwerfen Sie für den "Programmierknecht" eine **Beschreibung** (Leistungsbeschreibung) dieser Funktion mit dem im Unterricht verwendeten Schema. (Kein Programm !!!)

3)

12P

Betrachten Sie den folgenden Programmausschnitt:

```
...
int main() {
    float r;
    float u;
    r = 2;
    u = 3;
    → 1
    berechne_umfang (r, u);
    → 2
    printf("Radius= %f, Umfang= %f", r, u);
}

void berechne_umfang(float radius, float umfang) {
    umfang = 2 * 3.14 * radius ;
}
```

Durch die Deklaration der Variablen `r` und `u` werden die folgenden Zellen

	Adresse	Inhalt
<code>r</code>	0120	?
<code>u</code>	0130	?

im Arbeitsspeicher reserviert.

a) Welchen Wert hat der Inhalt der Variablen `r` und `u` an der Stelle 1 im Programm ?

b) Welchen Wert hat `radius` und `umfang` beim Aufruf von `berechne_umfang (r, u)` ?

c) Was bewirkt die folgende Anweisung im Arbeitsspeicher an der Adresse 0130 ?

`umfang = 2 * 3.14 * radius;`

d) Welchen Wert hat der Inhalt der Variablen `r` und `u` an der Stelle 2 im Programm ?

4)

13P

Programmieren Sie anhand der folgenden Beschreibung die dazugehörige Funktion:

```

/*****
/**
/**  int ersatz(double r1, double r2, int mod, double *rg)  **/
/**
/**
/**
/*#*****/
/*

```

Parameter:

```

(i) double r1>0:    erster Widerstandswert
(i) double r2>0:    zweiter Widerstandswert
(i) int mod:         10: Parallelschaltung
                    20: Reihenschaltung
(o) double *rg:     Gesamtwiderstand

```

Return:

```

(o) 0: Parallelschaltung oder Reihenschaltung wurde
    berechnet (mod ist 10 oder 20)
    -1: mod ist weder 10 noch 20

```

Beschreibung:

Berechnet in Abhängigkeit vom Modus mod (10 bedeutet eine Parallelschaltung, 20 bedeutet eine Reihenschaltung), den Widerstandswerten r1 und r2 den Ersatzwiderstand (Gesamtwiderstand) rg der Widerstandsschaltung.

\*/

## Lösungen

- 1) 12P  
a) zahl: 4, quad: 8 1P + 1P  
b) z: 4, zq: 04711 1P + 2P  
c) In den Inhalt der Adresse 04711 wird der Wert 16 geschrieben. 4P  
d) zahl: 4, quad: 16 1P + 2P

2) 13P  
/\*\*\*\*\*/  
/\*\* \*\*/  
/\*\* double tr (double z1, double z2, int mod) \*\*/  
/\*\* \*\*/  
/\*#\*\*\*\*\*/

Parameter:

- (i) double z1: erste Zahl
- (i) double z2!=0, wenn mod=4: zweite Zahl
- (i) int mod∈{1;2;3;4}:
  - 1: berechnet Summe z1+z2
  - 2: berechnet Differenz z1-z2
  - 3: berechnet Produkt z1\*z2
  - 4: berechnet Quotient z1/z2

Return:

- (o) Ergebnis der gewünschten Operation (Summe, oder Differenz oder Produkt oder Quotient).

Beschreibung:

Berechnet in Abhängigkeit vom Modus mod:

Summe z1+z2, wenn mod = 1

Differenz z1-z2, wenn mod = 2

Produkt z1\*z2, wenn mod = 3

Quotient z1/z2, wenn mod = 4

\*/

Jede fehlende Zusicherung: -2 P

z2!=0 ist keine richtige Zusicherung, da dann z.B. auch 3 \* 0 verboten würde: -2 P

Bemerkung zum Begriff Zusicherung:

Die Angabe beim Parameter z1:

z2!=0, wenn mod=4

und die Angabe:

mod∈{1;2;3;4}

nennt man Zusicherung. Das bedeutet, daß unter diesen Voraussetzungen der Programmierer dieser Funktion für die Korrektheit der Berechnungen dieser Funktion **garantiert**.

Je weniger Zusicherungen der Programmierer macht, desto größer wird der programmtechnische Aufwand für ihn, desto mehr "Intelligenz" muss er in die Funktion packen.

- 3) 12P
- a)  $r = 2, u = 3$  1P + 1P
- b) radius = 2, umfang = 3 1P + 2P
- c) nichts 4P
- d)  $r = 2, u = 3$  3P

4) 13P

```
int ersatz(double r1, double r2, int mod, double *rg){
    int r;
    if(mod==10){
        *rg=1/(1/r1+1/r2);
        r=0;
    }
    else if(mod==20){
        *rg=r1+r2;
        r=0;
    }
    else
        r=-1;
    return(r);
}
```