

KLAUSUR 1 Programmierpraktikum 2BK11 1.12.2016 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT 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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main()) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programnteile.
- 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 mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programnteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

1)

50P

Es soll ein Taschenrechner programmiert werden.

Zuerst muss dazu ein Zeichen über Tastatur eingegeben werden:

Bei Eingabe des Zeichens A oder a wird eine Addition durchgeführt,

bei Eingabe des Zeichens S oder s wird eine Subtraktion durchgeführt,

bei Eingabe des Zeichens M oder m wird eine Multiplikation durchgeführt,

bei Eingabe des Zeichens D oder d wird eine Division durchgeführt,

bei Eingabe eines anderen als der oben beschriebenen Zeichen, muß das Programm **sofort** beendet werden und eine entsprechende Meldung auf den Bildschirm ausgegeben werden. (insbesondere dürfen dann nicht mehr weitere Zahlen eingegeben bzw. etwas gerechnet werden).

Dann müssen - falls das Programm nicht beendet werden soll - 2 Zahlen eingegeben werden und die entsprechende Rechenoperation ausgeführt werden.

Bemerkungen:

1) Dies muß mit dem **EVA-Prinzip** realisiert werden.

2) Es widerspricht nicht dem EVA-Prinzip, wenn z.B. im Eingabeteil in bestimmten Variablen bestimmte Zustände abgespeichert werden (z.B: ob das Programm beendet werden soll) und diese dann im Ausgabeteil abgeprüft werden.

3) return darf nur einmal (am Ende des Programms) benutzt werden.

Programm darf nicht durch exit(...) oder sonstige Befehle verlassen bzw. beendet werden.

4) Durch 0 darf nicht dividiert werden.

Lösung

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

int main(){
    double zahl1, zahl2;
    double ergebnis;
    char zeichen;
    int zustand;
    // -1 : Programm beenden
    // -2 : Division durch 0
    // 0 : alles okay

    // E I N G A B E T E I L
    printf("Taschenrechner\n");
    printf("Addieren: A oder a eingeben \n");
    printf("Subtrahieren: S oder s eingeben \n");
    printf("Multiplikizieren: M oder m eingeben \n");
    printf("Dividieren: D oder d eingeben \n");
    printf("Programmende: irgendein anderes Zeichen eingeben \n");
    scanf("%c", &zeichen);

    if(zeichen=='A' || zeichen=='a' || zeichen=='S' || zeichen=='s'
        || zeichen=='M' || zeichen=='m' || zeichen=='D' || zeichen=='d')
        zustand = 0;
    else
        zustand = -1;

    if(zustand==0){
        printf("Bitte Zahl1 eingeben\n");
        scanf("%lf",&zahl1);
        fflush(stdin);

        printf("Bitte Zahl2 eingeben\n");
        scanf("%lf",&zahl2);
        fflush(stdin);
    }

    // V E R A R B E I T U N G S T E I L
    if(zustand == 0){
        if(zeichen=='A' || zeichen=='a'){
            ergebnis = zahl1 + zahl2;
        }

        else if(zeichen=='S' || zeichen=='s'){
            ergebnis = zahl1 - zahl2;
        }
        else if(zeichen=='M' || zeichen=='m'){
            ergebnis = zahl1 * zahl2;
        }
        else { // Division
            if(zahl2!=0){
                ergebnis = zahl1 / zahl2;
            }
            else{
                zustand = -2;
            }
        }
    }
}
```

```
// A U S G A B E T E I L
if(zustand==0){
    printf("Ergebnis=%f", ergebnis);
}
else if(zustand==-1){
    printf("Programmende\n");
}
else{ //Division durch 0
    printf("Durch 0 darf nicht dividiert werden\n");
}
}
return 0;
}
```

KLAUSUR 1 Programmierpraxis 2BKI1 Nachtermin 1 Zeit: 90 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT 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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main()) 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 mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

1) Schreiben Sie ein C-Programm, das die Vereinigungsmenge zweier Zahlenmengen berechnet:

Über Tastatur werden die zwei Elemente (müssen jeweils verschieden sein) der Menge A eingegeben. Dann werden über Tastatur die zwei Elemente (müssen jeweils verschieden sein) der Menge B eingegeben.

Wurden für eine Menge zwei gleiche Zahlen eingegeben, muß das Programm **sofort** beendet werden und eine entsprechende Meldung auf den Bildschirm ausgegeben werden.

(insbesondere darf dann nicht mehr eine weitere Zahl eingegeben und der Durchschnitt und die Vereinigung berechnet werden).

Dies muß mit dem EVA-Prinzip realisiert werden.

Außerdem muß in der Ausgabe noch angegeben werden, in welcher Menge

(z.B. 2. Menge) 2 gleiche Zahlen eingegeben wurde (und welcher Wert eingegeben wurde).

Erstellen Sie das dazugehörige C-Programm.

Bemerkungen:

a) Es widerspricht nicht dem EVA-Prinzip, wenn z.B. im Eingabeteil in bestimmten Variablen bestimmte Zustände abgespeichert werden (z.B: ob in der 1. Menge zwei gleiche Zahlen eingegeben wurden) und diese dann im Ausgabeteil abgeprüft werden.

b) Im Ergebnis darf ein Element auch nur einmal vorkommen, also z.B: {1; 2; 3} und nicht {1; 2; 3; 2}

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT 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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main()) 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 mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

AUFGABEN

1) 2P

Welchen Wert hat die Variable z nach Ausführung der folgenden C-Anweisung ?

Begründen Sie !

$z = 1/4 * 4;$

2) 2P

Wie viele Bit braucht man, um genau 32 verschiedene Zustände darzustellen ?

3) 6P

Stellen Sie den folgenden Pseudocode durch ein Struktogramm dar, wobei dort nur einseitige Verzweigungen vorkommen dürfen.

```
if (B) {  
    A1;  
}  
else {  
    A2;  
}
```

4) 5P

Annahme: Die Variable x soll vor der Ausführung folgender Anweisung nur die zwei verschiedenen Zahlenwerte A bzw. B annehmen können:

$x = A + B - x;$

a) Welche Werte kann dann x nach Ausführung dieser Anweisung annehmen ?

b) Wie kann man dann diese Anweisung mit Hilfe einer Verzweigung in C programmieren ?

5)

5P

a) Welchen Wert haben x und y nach Ausführung des folgenden Programms:
Geben Sie den Wert von x und y nach jeder Anweisung an.

```
...
x = 7;
y = 3;
x = x - y;
y = x + y;
x = y - x;
...
```

b) Was macht dieses Programm also im Endeffekt mit dem Inhalt der Variablen x und y (verbale Beschreibung) ?

6)

10P

Erstellen Sie ein Struktogramm, das das Minimum dreier Zahlen (Variablen z1, z2, z3) berechnet, in der Variablen min speichert und auf dem Bildschirm ausgibt. EVA-Prinzip beachten!

7)

10P

Der folgende syntaktisch korrekte Teil eines C-Programms soll die größte Zahl (dreier Zahlen z1, z2, z3) berechnen und in der Variablen max speichern.

Wenn das Programm korrekt ist, schreiben Sie "Der Algorithmus ist korrekt" und geben 5 Beispiele (mit konkreten Werten für z1, z2, z3 und dem berechneten Wert von max) an, wo er korrekt wird.

Wenn das Programm nicht korrekt ist, schreiben Sie "Der Algorithmus ist nicht korrekt" und geben ein **konkretes** Beispiel (mit konkreten Werten für z1, z2, z3 und dem berechneten Wert von max) an, bei dem der Algorithmus falsch wird.

```
... main(...) {
    ...
    max = z2;
    if (z1 < z2) {
        max = z1;
    }
    if (max < z3) {
        max = z3;
    }
    ...
}
```

8)

10P

Erstellen Sie ein Struktogramm zu dem Programm, das zu der in einer Prüfung erreichten Punktezahl p, (die der Lehrer über Tastatur eingibt), folgende Meldung ausgibt:

p zwischen 0 und 36 Punkte (je einschließlich):	"Prüfung nicht bestanden"
p größer 36 Punkte und kleiner (oder gleich) 100 Punkte:	"Prüfung bestanden"
p nicht im Bereich zwischen 0 und 100:	"unzulässige Punktezahl"

Bemerkung: Die Punktezahl kann auch nicht ganzzahlig sein!

Lösungen:

1) 2P
 $z = 0$, da 1 und 4 integer-Zahlen sind, deren Division den Wert 0 ergibt.

2) 2P
 5 Bit

3) 6P

W	B
A1	max = a
W	nicht B
A2	

4) 5P

a) 2P

A bzw. B

b) 3P

if(x==A)

 x = B;

else

 x = A;

5) 5P

a) 3P

x = 7;

y = 3;

x = x - y; // x = 7-3 = 4

y = x + y; // y = 4+3 = 7

x = y - x; // x = 7-4 = 3

b) 2P

Das Programm vertauscht den Wert von x und y

6) 10P

Eingabe(z1, z2, z3)	
W	$z1 < z2$
min = z1	min = z2
W	$z3 < \text{min}$
min = z3	
Ausgabe(min)	

7)

10P

Der Algorithmus ist nicht korrekt:

$z1 = 30 \quad z2 = 20 \quad z3 = 10 \implies \max = 20$

8)

10P

W		!(p>=0 && p<=100)		F			
Ausgabe(unzulässige Punktezahl)		W		p<=36		F	
		Ausgabe(Prüfung nicht bestanden)		Ausgabe(Prüfung bestanden)			

KLAUSUR 1 Programmiertheorie 2BKI1 Nachtermin 1 Zeit: 90 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT 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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main()) 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 mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vormane_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

AUFGABEN

1) 5P

- a) Wie viele Zustände genau kann man mit 8 Byte angeben? (als Hochzahl schreiben)
b) Näherungsweise Berechnung!

2) 10P

Simulieren Sie die folgende if-else Verzweigung durch eine oder mehrere einseitige Verzweigungen, wobei außer dem nicht Operator ! keine weiteren logischen Operatoren verwendet werden dürfen.

```
if (B1 && B2) {  
    A1  
}  
else{  
    A2  
}
```

3) 15P

Erstellen Sie ein Struktogramm, das drei Zahlen (Variablen z1, z2, z3) der Größe nach in aufsteigender Reihenfolge sortiert, in die Variablen klein, mittel, gross speichert und auf dem Bildschirm ausgibt. EVA-Prinzip beachten!

4)

20 P

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,

a) Erstellen Sie ein Flussdiagramm, das folgendes macht:

Es muß von einer eingegebenen ganzen Zahl $z \geq 2$ festgestellt werden, ob diese eine Primzahl ist. Das Ergebnis muß dann auf dem Bildschirm ausgegeben werden.

Tipp: Der Operator % berechnet den Rest bei einer Division. Damit kann man dann feststellen, ob eine Zahl eine andere teilt.

Beispiele:

$38 \% 3 = 2$, weil $38 : 3 = 12$ Rest 2, also teilt 3 nicht die Zahl 38

$38 \% 11 = 5$, weil $38 : 11 = 3$ Rest 5, also teilt 11 nicht die Zahl 38

$38 \% 2 = 0$, weil $38 : 2 = 19$ Rest 0, also teilt 2 die Zahl 38

KLAUSUR 2 Programmierpraktikum 2BK11 12.1.2017 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT 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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main()) 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 mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

1)

50P

Schreiben Sie ein Programm, das den Ersatzwiderstand (in Ohm) von den vom Anwender über Tastatur eingegebenen, parallel geschalteten Widerständen berechnet und auf dem Bildschirm (in Ohm) ausgibt.

Solange der Anwender einen negativen Widerstand eingibt, solange wird er aufgefordert, einen positiven Widerstand einzugeben.

Gibt der Anwender den Widerstandwert 0 ein, wird das Programm beendet und der Ersatzwiderstand aller bisher eingegeben Widerstände > 0 berechnet.

Bemerkung:

Für den Ersatzwiderstand r_{Ersatz} von n parallel geschalteten Widerständen r_1, \dots, r_n gilt:

$$\frac{1}{r_{\text{Ersatz}}} = \frac{1}{r_1} + \frac{1}{r_2} + \dots + \frac{1}{r_n}$$

Lösung:

```
#include "stdafx.h"

int main(int argc, char* argv[])
{
    double ersatz;
    double w;
    double summe=0;
    int beenden=0;

    do{
        printf("Bitte einen Widerstandswert in Ohm > 0  
eingeben\n");
        scanf("%lf",&w);
        if(w>0){
            summe=summe+1/w;
        }
        if(w==0){
            beenden=1;
        }

    }while(w!=0 && beenden==0);

    if(summe!=0)
        ersatz=1/summe;

    if(summe==0){
        printf("Sie haben keinen Widerstand > 0  
eingegeben\n");
    }
    else{
        printf("Ersatzwiderstand = %f\n",ersatz);
    }
    return 0;
}
```

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT 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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main()) 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 mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

AUFGABEN

1) 12P

Wie oft wird in den folgenden Programmausschnitten jeweils die Meldung "Hallo Welt" auf dem Bildschirm ausgegeben ?

a) 2P

```
x = 2;
while (x<=2) {
    printf("Hallo Welt\n");
}
```

b) 2P

```
do{
    x=3;
    printf("Hallo Welt\n");
}while(3<x);
```

c) 2P

```
while(5>5){
    printf("Hallo Welt\n");
}
```

d) 3P

```
int i;
for(i=0;i<101;i++){
    printf("Hallo Welt\n");
    i=100;
}
```

e) 3P

```
int erg=0;
int i=1;
while (erg==0) {
    i=i+1;
    erg=i%2;
    printf("Hallo Welt\n");
}
```

2)

6P

Simulieren Sie die Anweisung

```
if (B) {
    A
}
```

durch eine Schleife Ihrer Wahl. Dabei darf keine if-Anweisung vorkommen.

3)

6P

Ein Anwender soll eine ganze Zahl zwischen 1 und 5 (je einschließlich) über Tastatur eingeben. Schreiben Sie dazu einen Programmausschnitt in C, in der der Anwender gezwungen wird, so lange eine Zahl einzugeben, bis diese Bedingung erfüllt ist. Erst dann soll im Programm die nächste Anweisung erreicht werden.

4)

13P

Ein Anwender gibt über Tastatur ganze, positive Zahlen ≥ 0 ein. Wenn er das Programm beenden will, gibt er eine negative Zahl ein. Aus diesen ganze, positiver Zahlen muß das Maximum bestimmt und dann auf dem Bildschirm ausgegeben werden. Schreiben Sie das zugehörige Programm in C.

5)

14P

Das Programm (siehe unten) gibt etwas auf dem Bildschirm aus.

Die Tabelle zeigt einen 7 x 11 Bildschirm (7 Zeilen und 11 Spalten) an.

Tragen Sie die Ausgabe des Programms in die Tabelle (Bildschirm) ein.

Bem:

Schreiben Sie bei jedem Durchgang die neuen Werte der Variablen über die jeweilige Variable im Programm.


```
int main() {
    int i, j, k;
    for (i=0; i<4; i++) {
        for (j=0; j<i; j++) {
            printf(" ");
        }
        for (j=0; j<7-2*i; j++) {
            printf("#");
        }
        for (k=0; k<=i; k++) {
            printf(" ");
        }
        for (k=0; k<3; k++) {
            printf("#");
        }
        printf("\n");
    }
    return 0;
}
```

Lösungen:

1) 12P
unendlich, 1, 0, 1, 2

2) 6P
zustand=0;
while(B && zustand==0){
 A;
 zustand = 1;
}

```
// Alternative  
while(B){  
    A;  
    B = false;  
}
```

3) 6P
...
do {
 printf("ganze Zahl zwischen 1 und 5 eingeben\n");
 scanf("%d", &i);
}while(!(i>=1 && i<=5);

4) 13P
int main(){
 int zustand=0; //0 nicht beenden, -1 beenden
 int anzahl=0;
 int max=0;
 int zahl;
 while(zustand==0){
 printf("Positive Zahl eingeben oder mit negativer Zahl
 Programmende\n");
 scanf("%d",&zahl);

 if(zahl<0){
 if(anzahl==0){
 zustand=-2;
 }
 else{
 zustand=-1;
 }
 }
 else{
 zustand=0;
 }

 if(max<zahl){
 max=zahl;
 }
 anzahl++;
 }
 // Ausgabe
 if(zustand==-2){
 printf("existiert kein Maximum");
 }
 if(zustand==-1){
 printf("max=%d\n", max);
 }
 return 0;
}

5)

14P

[illegible]

Name, Vorname:

[illegible]

Lösungen:

```
#include "stdafx.h"

int main(){
    int i;
    int j;
    int anzahl;
    int summe=0;

    printf("Bitte anzahl, also Baumhoehe >= 1 eingeben\n");
    scanf("%d",&anzahl);

    for(i=0;i<anzahl;i++){
        for(j=0;j<anzahl-1-i;j++){
            printf(" ");
        }
        for(j=anzahl-1-i;j<=anzahl-1+i;j++){
            summe++;
            printf("*");
        }
        printf("\n");
    }
    printf("\n");
    printf("Summe aller Sterne=%d\n",summe);
    printf("\n\n\n");
    return 0;
}
```

KLAUSUR 3 Programmierpraktikum 2BK11 27.4.2017 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT 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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main()) 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 mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

Alle Teilaufgaben müssen in **einem** Programm realisiert werden.

AUFGABEN

1)

50P

Alle Felder dieses Programms sind Zeichenfelder.

Die Felder heißen feld1, feld2 und ergFeld.

ergFeld besteht aus der Aneinanderfügung der Zeichen von feld2 an feld1.

feld1 und feld2 haben jeweils die Länge LEN (mit Konstante arbeiten).

ergFeld hat die Länge 2*LEN (mit Konstante arbeiten).

Ausser LEN darf keine weitere Konstante benutzt werden!

(siehe Beispiel).

Beispiel: LEN = 10

feld1: Länge LEN

V	3	'\0'	?	?	?	?	?	?	?
---	---	------	---	---	---	---	---	---	---

feld2: Länge LEN

g	u	t	'\0'	?	?	?	?	?	?
---	---	---	------	---	---	---	---	---	---

ergFeld: Länge 2*LEN

V	3	g	u	t	'\0'	?	?	?	?	...	?
---	---	---	---	---	------	---	---	---	---	-----	---

- a) 3P
Alle Zellen des Feldes `ergFeld` müssen mit `'\0'` vorbelegt werden.
Implementieren Sie dazu den nötigen C-Quellcode.
- b) 4P
`feld1` muß mit `"V3"` und `feld2` mit `"gut"` durch die entsprechende Deklaration gleich mitinitialisiert werden.
Implementieren Sie dazu den nötigen C-Quellcode.
- c) 8P
Geben Sie den Inhalt der Felder `feld1` und `feld2` auf dem Bildschirm aus.
Bitte bei der Funktion `printf` kein `%s` benutzen.
Implementieren Sie dazu den nötigen C-Quellcode.
- d) 30P
`ergFeld` besteht aus der Aneinanderfügung der Zeichen von `feld2` an `feld1`.
Implementieren Sie dazu den nötigen C-Quellcode.
- e) 4P
Geben Sie den Inhalt des Felds `ergFeld` auf dem Bildschirm aus.
Implementieren Sie dazu den nötigen C-Quellcode.

Bemerkungen:

B1)

Das Programm muß mit einer beliebigen Vorbelegung von `feld1` und `feld2` funktionieren, nicht nur für `feld1` (mit Vorbelegung `"V3"`) und `feld2` (mit Vorbelegung `"gut"`),

B2)

Außer `printf(...)` bzw. `scanf(...)` dürfen keine andere Funktionen der Entwicklungsumgebung (wie z.B. `pow(...)`) benutzt werden).

Lösungen:

```
#include "stdafx.h"
```

```
#include "stdio.h"
```

```
const int LEN = 100; // 1P
```

```
int main()
```

```
{
```

```
    int i,j;
```

```
    char feld1[LEN]="Der_V3_ist_"; // 2P
```

```
    char feld2[LEN]="sehr_gut_und_liebt_alle"; // 2P
```

```
    char ergFeld[2*LEN]; // 2P
```

```
    for(i=0;i<2*LEN;i++){ // 3P
        ergFeld[i]='\0';
    }
```

```
    // 4P
```

```
    printf("\nInhalt von feld1: \n");
```

```
    for(i=0;feld1[i]!='\0';i++){
        printf("%c",feld1[i]);
    }
```

```
    // 4P
```

```
    printf("\nInhalt von feld2: \n");
```

```
    for(i=0;feld2[i]!='\0';i++){
        printf("%c",feld2[i]);
    }
```

```
    // 30P
```

```
// kopieren (5P)
```

```
    i=0;
```

```
    while(feld1[i]!='\0'){
        ergFeld[i]=feld1[i];
        i++;
    }
```

```
    j=0;
```

```
// anfügen
```

```
    while(feld2[j]!='\0'){
        ergFeld[i]=feld2[j];
        i++;
        j++;
    }
```

```
    ergFeld[i]='\0';
```

```
    // 4P
```

```
    printf("\nInhalt von ergfeld: \n");
```

```
    for(i=0;ergFeld[i]!='\0';i++){
        printf("%c",ergFeld[i]);
    }
```

```
    scanf("%d",&i);
```

```
    return 0;
```

```
}
```

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT 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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main()) 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 mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

AUFGABEN

1) 3P

In einem C-Programm wurde folgende Variable deklariert:

```
char v[4];
```

Geben Sie Ihren Kommentar zu folgenden Anweisungen ab:

```
v[0] = 'v';  
v[4] = 'r';  
v[-1] = 'r';
```

2) 3P

Wieviel Speicherplatz (in Bytes) benötigt der durch die folgende Deklaration reservierte Speicher ? Begründen Sie !

```
double werte[10][20];
```

3) 3P

Ein Programm fordert einen Anwender auf, in 100 verschiedenen Eingabefenstern Angaben (z.B. Name, Vorname, Wohnort, usw.) zu seiner Person (in Form von Zeichenketten) zu machen.

Der dafür reservierte Speicher soll vom Programmierer am Programmanfang mit '\0' initialisiert werden. Sollen für das Programm 100 eindimensionale oder ein zweidimensionales Feld verwendet werden ?

Beurteilen und begründen Sie dies aus der Sicht des Programmierers, der möglichst wenig Programmcode schreiben will.

4) 3P

Es wird das folgende zweidimensionale Feld deklariert:

```
char v[5][3];
```

Was ist (aus was besteht) v[1] ? Exakte Beschreibung !

5)

6P

Ein zweidimensionales Integer-Feld `feld2` besteht aus ZANZ Zeilen und SANZ Spalten (ZANZ und SANZ sind jeweils Konstanten). Dieses zweidimensionale Integer-Feld soll durch ein eindimensionales Integer-Feld `feld1` simuliert werden, in dem die Zeilen jeweils hintereinander angefügt werden.

Beispiel:

aus

1	2	3	4
3	5	7	9
2	4	6	8

wird:

1	2	3	4	3	5	7	9	2	4	6	8
---	---	---	---	---	---	---	---	---	---	---	---

Es gilt dann z.B:

`feld1[6] = feld2[1][2]`

Geben Sie allgemein an (auf das ZANZ x SANZ - Feld bezogen) an, durch welchen Term man den Platzhalter ? ersetzen muß?

`feld1[?] = feld2[i][j]`

6)

10P

Ein Text ist in einem zweidimensionalen Feld 3 x 3 Feld `ff` abgespeichert (siehe Beispiel). 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.

Der folgende syntaktisch korrekte Programmausschnitt soll dies bewerkstelligen.

```
int i, j;
for (i=0; i<3; i++) {
    for (k=0; k<3; k++) {
        ff[i][k] = ff[k][i];
    }
}
```

Nehmen Sie dazu Stellung.

Wenn dieses Programm nicht korrekt sein sollte, geben Sie bitte die Werte des Feldes `ff` nach Programmausführung an (auf das Beispiel bezogen).

a	b	c
d	e	f
g	h	i

→

a	d	g
b	e	h
c	f	i

7)

22P

Es soll ein C-Programm entwickelt werden, in dem ein Anwender die Widerstandswerte der Widerstände einer elektrischen Schaltung eingeben kann. Diese Widerstandswerte müssen in einem eindimensionalen Feld gespeichert werden.

Erzeugen Sie dazu ein Struktogramm !

Bemerkung:

B1) Überlegen Sie sich, wie Sie mit der Eingabe von negativen Widerstandswerten umgehen. (In der Elektrotechnik sind - zumindest hier für uns - nur positive (>0) Widerstandswerte physikalisch sinnvoll).

B2) Das Programm muß stabil sein, d.h. es darf z.B. nicht "abstürzen", weil der Anwender mehr Widerstandswerte eingegeben hat, als das Feld an Zellen besitzt.

Lösungen:

1)

3P

```
v[0] = 'v';    // richtig
v[4] = 'r';    // falsch: nicht reservierter Speicher
v[-1] = 'r';   // falsch: das erste Feld beginnt bei 0
```

2)

3P

$10 * 20 * \text{sizeof}(\text{double}) = 1600 \text{ Byte}$

3)

3P

Bei 100 eindimensionalen Feldern braucht man 100 Schleifen, dagegen ist bei einem zweidimensionalen Feld nur eine Schleife nötig.

4)

3P

v[1] ist ein eindimensionales Feld, das aus den folgenden Elementen besteht:
v[1][0], v[1][1], v[1][2]

5)

6P

Geben Sie an, durch welchen Term man den Platzhalter ? ersetzen muß?
`feld1[i*SANZ+j] = feld2[i][j]`

6)

10P

```
int i, k;
for (i=0; i<3; i++) {
    for (k=0; k<3; k++) {
        ff[i][k] = ff[k][i];
    }
}
```

a	b	c	→	a	d	g
d	e	f		d	e	h
g	h	i		g	h	i


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

void main(){
    const int len=5;
    double widerstaende[len];
    int anz = 0;
    double temp;
    // Schleifenabbruch: beenden=1, sonst beenden=0
    // weiter in der Schleife: beenden = 0
    int beenden = 0;

    printf("Bitte maximal %d Widerstaende eingeben\n", len-1);
    do{
        printf("Widerstaende eingeben\n");
        scanf("%lf", &temp);
        if(temp<=0){
            widerstaende[anz]=0;
            beenden = 1;
        }
        else{
            if(anz==len-2){
                widerstaende[anz]=temp;
                widerstaende[anz+1]=0;
                beenden=1;
            }
            else{
                widerstaende[anz]=temp;
                anz++;
                beenden=0;
            }
        }
    } while (beenden==0);
}
```

KLAUSUR 3 Programmierpraxis 2BKI1 Nachtermin 1 Zeit: 60 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT 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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main()) 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 mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vormane_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

AUFGABEN

- 1) 50 P
- 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,
Ein Primpaarzwilling sind 2 Primzahlen, deren Differenz 2 beträgt, wie z.B:
(3,5), (11,13), (17,19), ...

Schreiben Sie ein C-Program, das die ersten 100 Primpaarzwillinge auf dem Bildschirm ausgibt.

KLAUSUR 3 Programmiertheorie 2BKI1 Nachtermin 2 Zeit: 60 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT 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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main()) 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 mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vormane_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

AUFGABEN

1) 50 P

Gegeben ist das char-Feld feld1 der Länge LEN (Konstante), in dem Zeichen stehen (die z.B. ein Anwender eingegen hat).

In dieser Zeichenkette sollen alle Umlaute wie Ä, Ö, Ü, ä, ö, ü, ß durch entsprechende Doppelbuchstaben Ae, Oe,Ue, ae, oe, ue,ss ersetzt und im char-Feld feld2 der Länge 2*LEN abgespeichert werden.

Erstellen Sie dazu ein zugehöriges Struktogramm.

Beispiel:

feld1:

Tütensöhne

feld2:

Tuetensoehne

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT 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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main()) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programnteile.
- 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 mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programnteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

AUFGABEN

1) Es soll eine Zahlenfolge kodiert bzw. dekodiert werden.

Die Kodierung geschieht dadurch, dass zu jeder ganzen Zahl einer ganzzahligen Zahlenfolge eine bestimmte, konstante ganze Zahl dazu addiert wird.

Beispiel:

10, 7, -23, 19 ---+3---> 13, 10, -20, 22

Sie sollen dazu eine möglichst "luxuriös" gestaltete **Funktion** benutzen, die Sie aber wegen Arbeitsüberlastung im Fach Mathematik von einem "Programmierknecht" implementieren (programmieren) lassen.

- a) 20P
Entwerfen Sie für den "Programmierknecht" eine **Beschreibung** (Leistungsbeschreibung) der Funktion "kodieren" mit dem im Unterricht verwendeten Schema. (Kein Programm !!!)
Überlegen Sie sich zuerst genau, welche **Parameter** nötig sind
- b) 20P
Da sich der Programmierknecht zur Zeit in einem sogenannten "Tschill-Urlaub" befindet und deshalb aktuell (und wohl auch zukünftig) nicht ansprechbar ist, muss die Funktion "kodieren" von Ihnen selbst implementiert werden. Implementieren Sie die Funktion "kodieren".
- c) 5P
Schreiben Sie ein Programm mit einem Aufruf der Funktion "kodieren" (keine Eingabe über scanf(), sondern Aufruf mit konkreten Parametern).
- d) 5P
Rufen Sie die Funktion "kodieren" (mit geeigneten Parametern) direkt nach dem Aufruf in c) nochmals so auf, dass die veränderte Zahlenfolge wieder ihre ursprünglichen Werte bekommt (dekodieren).

Lösungen:

1a)

```
/* **** */
/**
/**  int kodieren (int zahlen[], int anzahl, int wert)  **/
/**                      // jeweils 2 P                **/
/*# **** */
/*
```

Parameter: // jeweils 3P

(i/o) int zahlen[]: zu kodierende Zahlenenfolge

(i) int anzahl: Anzahl der Zahlen der Zahlenenfolge

(i) int wert: Zahl, die zu der Zahlenfolge dazuaddiert wird

Return:

kein

Beschreibung: // 5P

Kodiert "anzahl" Zahlen der Zahlenfolge "zahlen", indem der Wert "wert" dazuaddiert wird.

Beispiel:

zahlen[5]: {1, 2, 3, 5, 6}

anzahl: 3

wert: 100

Nach dem Aufruf:

zahlen[5]: {101, 202, 203, 5, 6}

*/

b)

```
#include "stdafx.h"
```

```
void kodieren(int zahlen[], int anzahl, int wert);
```

```
int main(int argc, char* argv){
```

```
    int i;
    int zahlen[5]={1, 2, 3, 5, 6};
    kodieren(zahlen, 4, 100);
    for(i=0; i<4; i++){
        printf("%d ", zahlen[i]);
    }
```

```
    kodieren(zahlen, 4, -100);
    printf("\n");
    for(i=0; i<4; i++){
        printf("%d ", zahlen[i]);
    }
    return 0;
}
```

```
void kodieren(int zahlen[], int anzahl, int wert){
```

```
    int i;

    for(i=0; i<anzahl; i++){
        zahlen[i]=zahlen[i]+wert;
    }
}
```

KLAUSUR 4 Programmierpraktikum 2BK11 29.6.2017 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT 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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main()) 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 mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

1)

In einem aus Zeichen bestehenden Eingangsfeld (Array) soll das Auftreten eines jeden Zeichens verdoppelt werden.

Diese Veränderungen dürfen aber nicht im Eingangsfeld gemacht werden.

Beispiel: In der Zeichenfolge "V3gut" jedes Zeichen verdoppeln.

Eingangsfeld (Array)	Ausgangsfeld (Array)
V3gut	VV33ggguutt

a)

20P

Machen Sie zuerst eine Leistungsbeschreibung (Dokumentation) dieser Funktion verdoppleZeichen(...) nach dem im Unterricht verwendeten Schema.

b)

20P

Implementieren Sie die Funktion "verdoppleZeichen(...)"

c)

10P

In main muß ein Aufruf der Funktion "verdoppleZeichen(...)" realisiert werden.
(keine Eingabe über scanf(), sondern Aufruf mit konkreten Parametern).

Testen Sie diese Funktion, indem die Elemente des Ausgangsfeldes (Array) auf dem Bildschirm ausgegeben werden.

Beispiel einer Dokumentation einer Funktion:

```

/*****
/**
/**  int berechne(double a, double b, double *u, double *f)  **/
/**
/**#*****/
/*
Parameter:
    (i) double a>=0 :   eine Seitenlänge des Rechtecks
    (i) double b>=0 :   andere Seitenlänge des Rechtecks
    (o) double *u    :   Umfang des Rechtecks
    (o) double *f     :   Fläche des Rechtecks

Return:
    (o) 1:  Quadrat
        2:  Rechteck

Beschreibung:
    Berechnet aus den beiden positiven Seitenlängen a und b des Rechtecks
    den Umfang *u und die Fläche *f und bestimmt außerdem, ob es
    ein Quadrat oder ein Rechteck ist.

Beispiel:
    a:10 , b:20
    erg=berechne(a, b, &umfang, &flaeche)
    umfang:60 , flaeche:200 , erg:2
*/

int berechne(double a, double b, double *u, double *f){
    int r;
    *u=2*(a+b);
    *f=a*b;
    if(a==b){
        r=1;
    }
    else{
        r=2;
    }
    return(r);
}

```

```
#include "stdafx.h"
#include "stdio.h"
#include "malloc.h"
```

```
int main()
{
    int z;
    char stringIn[]="abcxyzx";
    char stringOut[15];
    verdoppleZeichen(stringIn, stringOut);
    printf("%s\n",stringIn);
    printf("%s\n",stringOut);
    scanf("%d",&z);
    return 0;
}
```

Parameter:

```
Return:
    nichts
```

Die Zeichenfolge, die sich in "stringIn" befinden, wird verdoppelt. Diese Aktion wird aber nicht in "stringIn" gemacht, sondern das Ergebnis nach "stringOut" kopiert. Jedes Element aus "stringIn" wird zwei Mal hintereinander in "stringOut" kopiert. Zur Sicherheit muß stringOut mindestens die doppelte Länge von stringIn haben plus 1, also:

```
len(stringOut) = 2*len(stringIn) + 1
```

```
void verdoppleZeichen(char stringIn[], char stringOut[]){
    int anz;
    int i,j;
    int len;
    int erg;
    erg=0;

    i=0;
    j=0;
    for(i=0; stringIn[i]!='\0'; i++){
        stringOut[j]=stringIn[i];
        j++;
        stringOut[j]=stringIn[i];
        j++;
    }

    stringOut[j]='\0';
    return;
}
```