

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) 2P

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

Begründen Sie !

`z = 1 / 4 * 4 ;`

2) (1P + 2P + 2P + 1P) 6P

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

b) Der Datentyp unsigned int besitzt 4 Byte und kann nur positive ganze Zahlen abspeichern.

Wie viele verschiedene Zahlen können damit abgespeichert werden ?

(Ergebnis mit Hilfe einer Hochzahl angeben und dann noch näherungsweise Berechnung!)

c) Wie heißt die größte (höchste) Zahl, die mit unsigned int abgespeichert werden kann ?

(Ergebnis binär angeben und dann Ergebnis mit Hilfe einer Hochzahl angeben!)

d) Zu dieser größten (höchsten) Zahl wird 1 dazuaddiert und in der Variablen unsigned int m gespeichert. Was gibt die folgende, syntaktisch korrekte Anweisung auf dem Bildschirm aus ?

(mit Begründung!)

`printf ("%u", m) ;`

3) 3P

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

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

4) 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!

5)

5P

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

$x = A + B - x;$

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

b) Durch welche Verzweigung kann man diese obige Anweisung gleichwertig ersetzen ?

Bitte Programmteil in C angeben.

6)

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) ?

7)

10P

Der folgende syntaktisch korrekte Teil eines C-Programms soll die größte Zahl dreier (in z1, z2, z3 gespeicherter) Zahlen 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) 6P

a) 5 Bit (1P)

b) $2^{32} = 2^{30+2} = 2^{30+2} * 2^2 = 2^{30} * 4 = (2^{10})^3 * 4 \approx (10^3)^3 * 4 \approx 10^9 * 4$ (2P)

c) $11111111 \ 11111111 \ 11111111 \ 11111111 = 2^{32} - 1$ (2P)

d) wenn man dazu 1 addiert, geht der Übertrag verloren und es bleibt: (1P)

$00000000 \ 00000000 \ 00000000 \ 00000000 = 0$

Es wird also 0 auf dem Bildschirm ausgegeben.

3) 3P

W	B
A1	
W	nicht B
A2	

4) 10P

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

5) 5P

a) 2P

A bzw. B

b) 3P

```
if(x==A){
```

```
    x = B;
```

```
}
```

```
else{
```

```
    x = A;
```

```
}
```

6)

5P

a) 3P

$x = 7;$

$y = 3;$

$x = x - y; \quad // \quad x = 7 - 3 = 4$

$y = x + y; \quad // \quad y = 4 + 3 = 7$

$x = y - x; \quad // \quad x = 7 - 4 = 3$

b) 2P

Das Programm vertauscht den Wert von x und y

7)

10P

Der Algorithmus ist nicht korrekt:

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

8)

10P

$!(p > 0 \ \&\& \ p \leq 100)$		
W	F	
Ausgabe(unzulässige Punktezahl)	$p \leq 36$	
	W	F
	Ausgabe(Prüfung nicht bestanden)	Ausgabe(Prüfung bestanden)

KLAUSUR 1 Programmierpraktikum 2BK11 23.11.2018 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_Rechnummer_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_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) 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

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) 4P

- a) Was bedeutet fußgesteuerte Schleife und wie wird diese auch noch genannt?
b) Was bedeutet kopfgesteuerte Schleife und wie wird diese auch noch genannt?

2) 4P

Erstellen Sie das zu der folgenden Anweisung gehörige Flußdiagramm:

```
do{  
    A;  
}  
while (B);
```

3) 12P

- a) Erstellen Sie einen Programmausschnitt einer while-Schleife und einer do-while-Schleife, die jeweils den gleichen Schleifenkörper, die gleiche Schleifenbedingung und die gleichen Anweisungen vor der Schleife haben, deren Schleifenrumpf aber gleich oft durchlaufen werden
b) Erstellen Sie einen Programmausschnitt einer while-Schleife und einer do-while-Schleife, die jeweils den gleichen Schleifenkörper, die gleiche Schleifenbedingung und die gleichen Anweisungen vor der Schleife haben, deren Schleifenrumpf aber verschieden oft durchlaufen werden

4) 3P

Wie viele Ausgaben auf dem Bildschirm erzeugt der folgende syntaktisch korrekte Programmausschnitt? Begründen Sie.

```
i = 100;  
while(i!=5) {  
    i = i-2;  
    printf("%d\n", i);  
}
```

5)

4P

Wie viele Ausgaben auf dem Bildschirm erzeugt der folgende syntaktisch korrekte Programmausschnitt? Begründen Sie.

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

6)

3P

Wie viele Ausgaben auf dem Bildschirm erzeugt der folgende syntaktisch korrekte Programmausschnitt? Begründen Sie.

```
i = 100;
do{
    i = i-1;
    i = i+2;
    printf("%d\n", i);
}while (i>101);
```

7)

10P

Es sei a eine Fließkommazahl und n eine ganze Zahl.
 a^n ist wie folgt definiert:

$$a^n = \begin{cases} a * a * \dots * a & \text{(n-mal } a \text{ mit sich selbst multipliziert) , falls } n > 0 \\ 1 & \text{falls } n = 0 \\ 1 / (a * a * \dots * a) & \text{(-n-mal } a \text{ mit sich selbst multipliziert), falls } n < 0 \end{cases}$$

Implementieren Sie einen Programmausschnitt, der a^n berechnet.

Beispiele: $a^3 = a * a * a$ $a^{-4} = 1 / (a * a * a * a)$ $a^0 = 1$

8)

10P

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

Die Tabelle zeigt einen 7 x 10 Bildschirm (7 Zeilen und 10 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;
    i=0;
    while(i<3){
        for(j=0;j<3-i;j++){
            printf(" ");
        }

        for(k=0;k<2*i+1;k++){
            printf("*");
        }
        printf("\n");

        i=i+1;
    }
    return 0;
}
```


Lösungen:

1)

4P

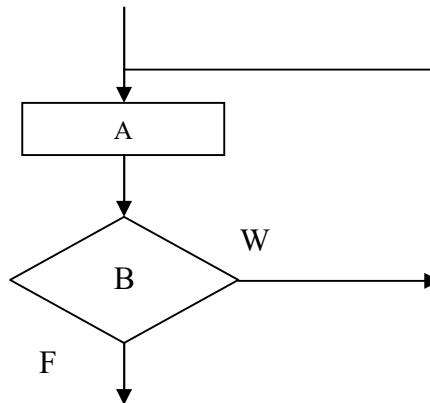
fußgesteuerte Schleife: Bedingung am Ende --> do-while-Schleife

kopfgesteuerte Schleife : Bedingung am Anfang --> while-Schleife

2)

4P

Erstellen Sie das zu der folgenden Anweisung gehörige Flußdiagramm:



3)

12P

a)

```
i=1;
while (i==1) {
    i=i+1;
}
```

```
i=1;
do{
    i=i+1;
}
while (i==1) ;
```

b)

```
while (false) {
    i = 1;
}
```

```
do{
    i = 1;
} while (false)
```

4)

3P

Unendlich viel Ausgaben

Da der Wert von i immer gerade ist, wird die Bedingung nie falsch.

5)

4P

Ein Mal, weil

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

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

6)

3P

Es wird eine Ausgabe auf dem Bildschirm erzeugt.

Da i nach dem 1. Durchgang den Wert 101 hat, wird die Bedingung falsch.

7)

```

...
double a;
double erg;
int n;

erg=1;

if(n>0){
    for(i=0;i<n;i++){
        erg=erg*a;
    }
}
if(n==0){
    erg=1;
}
if(n>0){
    for(i=0;i<n;i++){
        erg=erg*a;
    }
    erg=1/erg;
}

```

8)

[illegible]

KLAUSUR 2 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 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)

50P

Entwickeln Sie ein Programm für einen Tilgungsplan:

Nach Angabe von Kreditsumme, Darlehnszinsen pro Jahr (umrechnen in Darlehnszinsen pro Monat, siehe Rechenbeispiel unten) und monatlicher Rate sollen die Dauer der Tilgung und die Gesamtkosten (Summe aller Zinsen, die an die Bank gezahlt wurden !) berechnet werden. Die monatliche Rate wird zur Zinszahlung und Tilgung benutzt.

Mathematische Anleitung anhand eines Beispiels:

Voraussetzungen: Kreditsumme $S = 1000$ EURO, Darlehnszinsen pro Jahr $p = 120\%$, d.h. $120/12 = 10\%$ Darlehnszinsen pro Monat, monatliche Rate = 500 EURO.

n	Zinsen z_n	Tilgung tg_n	Schulden S_n
0	0	0	1000
1	$1000 \cdot 0,1 = 100$	$500 - 100 = 400$	$1000 - 400 = 600$
2	$600 \cdot 0,1 = 60$	$500 - 60 = 440$	$600 - 440 = 160$
3	$160 \cdot 0,1 = 16$	160	0

Im 3. Monat tilgt der Kunde 160 Euro und zahlt noch 16 Euro Zinsen. Deshalb bekommt er noch $500 - (16 + 160) = 324$ Euro zurück.

Insgesamt hat man an die Bank $100 + 60 + 16 = 176$ Euro Zinsen gezahlt.

KLAUSUR 2 Programmierpraktikum 2BK11 14.12.2018 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 !!!

AUFGABEN

1)

50P

Schreiben Sie ein C-Programm, das den Mittelwert von den vom Anwender über Tastatur eingegebenen Schulnoten berechnet und auf dem Bildschirm ausgibt.
Solange der Anwender eine Zahl eingibt, die eine Schulnote ist, wird er aufgefordert eine neue Note einzugeben.
Wenn der Anwender eine Zahl eingibt, die keine Schulnote ist, wird der Mittelwert aller bisher eingegeben Noten berechnet und auf dem Bildschirm ausgegeben..

Bemerkung:

1)

Da hier das EVA-Prinzip nicht eingehalten werden kann, muß zumindest der Ausgabe-Teil vom EV-Teil getrennt sein. Ausgabe-Teil durch Kommentar angeben.

2)

Am Anfang des Programms muss für den Anwender eine vollständige Programminformation (Programminfo) auf dem Bildschirm ausgegeben werden, in der beschrieben steht, wie der Anwender das Programm zu bedienen hat und was das Programm macht (z.B. kann dies durch die Angabe eines Beispiels ergänzt werden).

Lösung:

```
int main(){
    double note;
    int anzahl=0;
    double mittelwert=-1;
    double summe=0;
    int istNote=1;

    while(istNote==1){
        anzahl=anzahl+1;
        printf("Bitte Note eingeben\n");
        scanf("%lf",&note);
        if(note<1 || note >6){
            istNote=0;
        }
        else{
            summe=summe+note;
            mittelwert=summe/anzahl;
        }
    }

    // --- Ausgabe ---
    if(mittelwert==-1){
        printf("es gibt keinen Mittelwert, da keine Note eingegeben\n");
    }
    else{
        printf("Mittelwert=%lf\n",mittelwert);
    }

    return 0;
}
```

KLAUSUR 2 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_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

a) Nach der Eingabe der Zeilenzahl "anzahl" soll eine zweidimensionale Pyramide aus Sternen (*) auf dem Bildschirm ausgegeben werden, die aus "anzahl" Schichten besteht. Außerdem soll die Anzahl der Sterne der Pyramide berechnet und auf dem Bildschirm ausgegeben werden.

Beispiel (anzahl = 5)

```
* * * * *
 * * * *
  * * *
   * *
    *
```

b) Erweitern Sie das Programm so, dass das auszugebende Zeichen frei wählbar ist und zusätzlich die Form Doppelpyramide statt Pyramide eingegeben werden kann.

Beispiel (Anzahl = 4, Zeichen = '#')

```
#####
####
###
#
###
####
#####
```

KLAUSUR 3 Programmierpraktikum 2BK11 5.4.2019 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 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 3 Programmiertheorie 2BKI1 Nachtermin 1 Zeit: 80 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)

a)

40P

Eines morgens in aller Frühe finden Sie auf Ihrem Schreibtisch die folgende Leistungsbeschreibung (Dokumentation) der Funktion "ersetzen (...)" vor (siehe Rückseite). Implementieren Sie diese Funktion.
Wo nötig (bzw. von Vorteil) den Bezeichner const verwenden.

b)

5P

Schreiben Sie ein Programm, in dem ein Aufruf der Funktion "ersetzen (...)" verwendet wird (keine Eingabe über scanf(), sondern Aufruf mit konkreten Parametern). Testen Sie diese Funktion, indem die Elemente der neuen Folge auf dem Bildschirm ausgegeben werden.

c)

5P

Welchen Nachteil hätte es, wenn man in der Funktion ersetze(...) dynamisch Speicherplatz für "neueFolge" belegt?
Man könnte ja die Größe dieses Speicherplatzes innerhalb der Funktion ersetze(...) berechnen lassen und dann genau so viel Speicherplatz für "neueFolge" allokalieren, wie "neueFolge" benötigt.

```

/*****
/**
/** void ersetzen(char zeichen, char folge[],
/** char ersetzung[], char neueFolge[])
/**
/**
/*****
/*

```

Parameter:

- (i) char zeichen : zu ersetzendes Zeichen.
- (i) char folge[] : dort wird das "zeichen" ersetzt.
- (i) char ersetzung[]: Das Zeichen "zeichen" wird durch die Zeichenfolge "ersetzung" ersetzt.
- (o) char neueFolge[] : Die durch die Ersetzung entstehende neue Zeichenfolge

Return:

kein

Beschreibung:

In der Zeichenfolge "folge" wird beim erstmaligen Auftauchen des Zeichens "zeichen" dieses Zeichen durch die Zeichenfolge "ersetzung" ersetzt. Dadurch entsteht die neue Zeichenfolge "neueFolge".

Der Programmierer, der diese Funktion benutzt muß dafür Sorge tragen, dass beim Aufruf dieser Funktion genügend Speicherplatz für "neueFolge" bereitgestellt wird.

Beispiel 1:

zeichen: a
folge: "raav"
ersetzung: "xy"
Nach dem Aufruf:
neueFolge: "rxyav"

Beispiel 2:

zeichen: s
folge: "raav"
ersetzung: "xy"
Nach dem Aufruf:
neueFolge: "raav"

*/

Lösungen:

```
#include "stdafx.h"
#include <string.h>
#include <malloc.h>

void ersetzen(const char zeichen, const char folge[],
              const char ersetzung[], char neueFolge[]);

int main(int argc, char* argv[]){
    char folge[4]="raf";
    char ersetzung[3]="xy";
    char neueFolge[4];
    char zeichen='a';

    ersetzen(zeichen, folge, ersetzung, neueFolge);
    printf("neueFolge=%s\n",neueFolge);
    return 0;
}

void ersetzen(const char zeichen, const char folge[],
              const char ersetzung[], char neueFolge[]){
    int i=0;
    int j=0;
    int index=0;

    // Kopiere alle Elemente von folge bis zum Auftreten
    // des gefundenen Zeichens in neueFolge
    while(folge[i]!='\0' && folge[i]!=zeichen){
        neueFolge[i]=folge[i];
        i++;
    }
    // Zeichen wurde nicht gefunden
    if(i==strlen(folge)){
    }
    else{
        index=i;
        // Füge ersetzung an neueFolge an
        while(ersetzung[j]!='\0'){
            neueFolge[i]=ersetzung[j];
            i++;
            j++;
        }
        j = index+1;
        // Füge von j=index an die folge an die neue Folge an.
        while(folge[j]!='\0'){
            neueFolge[i]=folge[j];
            i++;
            j++;
        }
    }
    neueFolge[i]='\0';
}
```

KLAUSUR 4 Programmierpraktikum 2BK11 7.6.2019 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)

In einem aus Integer-Werten bestehenden Feld (Array) soll das erste Auftreten eines Integer-Werts gelöscht werden.

Diese Veränderungen müssen im gleichen Feld gemacht werden.

1. Beispiel:

In der Zahlenfolge 8, 9, 7, 3, 9, 7, 9, 8 das erstmalige Auftreten der Zahl 9 löschen.

Feld vor dem Aufruf: 8, 9, 7, 3, 9, 7, 9, 8

zu löschende Zahl: 9

Feld nach dem Aufruf: 8, 7, 3, 9, 7, 9, 8

2. Beispiel:

In der Zahlenfolge 8, 9, 7, 3, 9, 7, 9, 8 das erstmalige Auftreten der Zahl 2 löschen.

Feld vor dem Aufruf: 8, 9, 7, 3, 9, 7, 9, 8

zu löschende Zahl: 2

Feld nach dem Aufruf: 8, 9, 7, 3, 9, 7, 9, 8

Tipp:

Geben Sie die neue Feldlänge mit return zurück!

Die neue Feldlänge ist gleich der alten Feldlänge oder diese um 1 verringert.

a) 20P
Machen Sie zuerst eine Leistungsbeschreibung (Dokumentation) der Funktion `entferneZahl (...)` nach dem im Unterricht verwendeten Schema.

b) 20P
Implementieren Sie die Funktion "`entferneZahl (...)`" auf eine "luxuriöse" Art.
Dazu gehört, dass diese Funktion nicht nur die Zahl 9 eines Zahlenfeldes entfernen kann, sondern die Funktionalität durch die Parameter der Funktion möglichst allgemein realisiert werden kann!

c) 10P
In `main` muß ein Aufruf der Funktion "`entferneZahl (...)`" 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.

Bemerkungen:

1)
Außer `printf(...)` bzw. `scanf(...)` dürfen keine andere Funktionen der Entwicklungsumgebung (wie z.B. `pow(...)`) benutzt werden).

2)
Eventuell Vorbedingungen und Zusicherungen beachten.

Lösungen:

a)

```
/*
**
**  int entferneZahl(int feld[], int len, int zahl)
**
**
*/
/*#
**
**
*/
```

Parameter:

(i/o) int feld[] : Feld
(i) int len >0 : Länge des Feldes feld, genauer:
Anzahl der zu berücksichtigenden Elemente
des Feldes (vom Feldanfang gezählt).
(i) int zahl : Zu löschende Zahl im Feld

Return:

Länge des Feldes.
Falls die Zahl gefunden wird, wird ist die Felddlänge um 1
kleiner, also vor dem Aufruf. Sonst ist sie gleich.

Beschreibung:

Im Feld "feld" wird die Zahl "zahl" gesucht. Falls sie
gefunden wird, wird in "feld" ihr erstes Auftreten gelöscht.
Dadurch wird die Felddlänge um 1 verringert.
Wenn die Zahl nicht gefunden wird, bleibt das Feld
unverändert. Auch die Felddlänge ändert sich nicht

Beispiel 1:

```
int feld[10] = {1,2,3,1,10,11,12}
erg=entferneZahl(feld, 4, 1);
feld[] : {2,3,1}
erg : 3
```

Beispiel 2:

```
int feld[10] = {1,2,3,1,10,11,12}
erg=entferneZahl(feld, 4, 123);
feld[] : {1,2,3,1}
erg : 4
```

b)

```
int main()
{
    int newLen;
    int myFeld[100]={1,2,3,1,10,11,12};

    newLen = entferneZahl(myFeld, 4, 1);

    for(i=0;i< newLen;i++){
        printf("%d",myFeld[i]);
    }
    return 0;
}
```

c)

```
int entferneZahl(int feld[],int len , int zahl){
    int i=0;
    int index=-1;
    int lenNeu;
    // suche erstes Auftreten der "zahl"
    for(i=0;i<len;i++){
        if(feld[i]==zahl){
            index=i;
            break;
        }
    }
    // Zahl nicht gefunden
    if(index==-1){
        lenNeu=len;
    }
    else{
        lenNeu=len-1;
        if(index==len-1 || len==1){
            // es muss nichts entfernt werden
        }
        else{
            for(i=index+1;i<len;i++){
                feld[i-1]=feld[i];
            }
        }
    }
    return lenNeu;
}
```

Fehlerbewertungen:

- 1) In der Beschreibung steht bei return etwas anderes als was die Funktion zurückgibt: -5P
- 2) In der Beschreibung steht bei return eine Variable: -2P
- 3) In der Beschreibung wird das mögliche Nichtvorkommen von "zahl" in " feld" nicht beachtet: -3P
- 4) Inkonsistente Beschreibung: -10P

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) 50P

Im Fach Mathematik soll eine "Polynomfunktion n-ten Grades" mit dem Funktionsterm:

$$y = a_0 + a_1x^1 + a_2x^2 + \dots + a_nx^n$$

erstellt werden, die den Funktionswert (y-Wert) eines beliebigen x-Werts berechnet.

Der Mathelehrer gibt diese Aufgabe an die EDV-Abteilung weiter.

a) 20P

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

Diese Funktion fPolynom(...) berechnet den Funktionswert (y-Wert) einer Polynomfunktion n-ten Grades an einem beliebigen x-Wert.

Nochmals: Dies soll auf eine "luxuriöse" Art und Weise gemacht werden.

Dazu gehört, dass diese Funktion nicht nur den y-Wert eines Polynoms 3. Grades an der x-Stelle 2 berechnen kann, sondern die Funktionalität durch die Parameter der Funktion möglichst allgemein realisiert werden kann!

Tipp: Die Koeffizienten a_0, a_1, \dots, a^n in einem Feld speichern.

b) 20P

Implementieren Sie diese Funktion fPolynom(...).

c) 10P

In main muß ein Aufruf der Funktion "fPolynom" realisiert werden.

(keine Eingabe über scanf(), sondern Aufruf mit konkreten Parametern).

Konkret:

c1) Rufen Sie die Funktion fPolynom(...) mit den entsprechenden Parametern so in main auf, dass der y-Wert einer Normalparabel an der x-Stelle 4 berechnet wird, also der y-Wert gleich $1 * 4^2$ berechnet wird und geben Sie das Ergebnis auf dem Bildschirm aus.

c2) Geben Sie die Koeffizienten des Feldes bei c1) mit einer Schleife auf dem Bildschirm aus.

Bemerkungen:

1)

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

2)

Eventuell Vorbedingungen und Zusicherungen beachten.

Lösung:

1a)

```
/* **** */
/**
/**  double fPolynom(double koeffizienten [], int grad,
/**      double x)
/**
/**
/*# **** */
/*
```

Parameter:

(i) double koeffizienten[] : Feld mit Koeffizienten des Polynoms

(i) int grad >0 : Grad (höchste Potenz) des Polynoms.
Die Länge des Feldes koeffizienten, in dem die Koeffizienten gespeichert werden muss \geq grad+1 sein.
Es werden grad+1 Elemente des Feldes (vom Feldanfang gezählt) berücksichtigt.

(i) double zahl : x-Wert des Polynoms

Return:

koeffizienten[0]+koeffizienten[1]*x¹+
koeffizienten[2]*x²+...+koeffizienten[grad]*x^(grad)

Beschreibung:

Berechnet den y-Wert einer Parabel mit der Funktionsgleichung
koeffizienten[0]+koeffizienten[1]*x¹+
koeffizienten[2]*x²+...+koeffizienten[grad]*x^(grad)
an der Stelle x.

Beispiel 1:

```
double koeffizienten[10] = {1,2,3,1,10,11,12}
erg= fPolynom(koeffizienten, 2, 4);
erg : 57 =1 + 2*41 + 3*42
```

1b)

```
double fPolynom(double koeffizienten[], int grad, double x){
    int i;
    double summe=koeffizienten[0];
    double produkt=x;
    for(i=1;i<grad;i++){
        summe=summe+koeffizienten[i]*produkt;
        produkt=produkt*x;
    }
    return summe;
}
```

c)

```
int main(){
    int i;
    double erg;
    double myKoeffizienten[100]={0,0,1};

    erg=fPolynom(myKoeffizienten, 2, 4);
    printf("=%lf\n",erg);

    printf("Koeffizienten des Polynoms=");
    for(i=0;i<=2;i++){
        printf("%lf, ",myKoeffizienten[i]);
    }

    return 0;
}
```

oder ausführlicher:

```
int main(){
    int i;
    double erg;
    double zahl=4;
    int grad=2;
    double myKoeffizienten[100]={1,2,3};
    erg=fPolynom(myKoeffizienten, grad, zahl);

    printf("Koeffizienten des Polynoms=");
    for(i=0;i<=grad;i++){
        printf("%lf, ",myKoeffizienten[i]);
    }
    printf("\n");
    printf("f(%lf)=", zahl);
    printf("%lf+",myKoeffizienten[0]);
    for(i=1;i<=grad;i++){
        if(i<grad-1){
            printf("%lf*%lf^%d+",myKoeffizienten[i],zahl,i);
        }
        else{
            printf("%lf*%lf^%d",myKoeffizienten[i],zahl,i);
        }
    }

    printf("=%lf\n",erg);
    return 0;
}
```

KLAUSUR 4 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.

Lösungen:

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

```
bool istPrimzahl(int zahl){
    bool b;
    int i;
    int zaehler=0;
    for(i=1;i<=zahl;i++){
        if(zahl%i==0){
            zaehler++;
        }
    }
    if(zaehler==2){
        b=true;
    }
    else{
        b=false;
    }
    return b;
}
```

```
int main()
{
    int zahl=3;
    int erg=0;
    int zaehler=1;
    while(zaehler<=100){
        if(istPrimzahl(zahl) && istPrimzahl(zahl+2)){
            printf("(%d,%d) ", zahl,zahl+2);
            zaehler++;
        }
        zahl=zahl+2;
    }
    return 0;
}
```