

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.

AUFGABEN

1) 3+3+3 P

- a) Welche 3 Möglichkeiten (3 Worte nennen) gibt es, einen Algorithmus darzustellen?
- b) Erklären Sie die Begriffe Syntax und Semantik?
- c) Was ist ein Compiler ?

2) 5P

Der folgende Teil eines C-Programms soll die kleinste zweier in z1 und z2 gespeicherten Zahlen berechnen und in der Variable min abspeichern.

- a) Ist das Teilprogramm syntaktisch korrekt? Bitte begründen !
- b) Angenommen das Programm wäre syntaktisch korrekt. Ist das Programm auch semantisch korrekt (d.h. berechnet es auch das Minimum)?

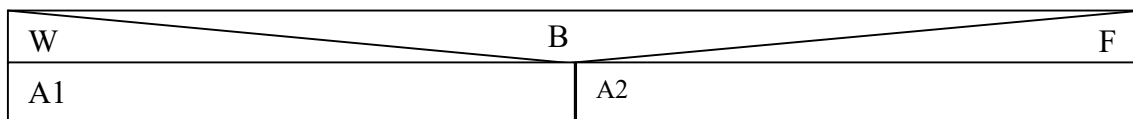
Wenn das Programm korrekt ist, schreiben Sie "Der Algorithmus ist korrekt".

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

```
...  
min = z1;  
if (z1<z2)  
    min=z1;  
...
```

3) 3P

Stellen Sie das folgende Struktogramm nur mit Hilfe von einseitigen Verzweigung dar (das die gleiche Semantik hat).



4) 5 P
a) Wie viele Zustände genau kann man mit 8 Byte angeben? (als Hochzahl schreiben)
b) Näherungsweise Berechnung!

5) 10P
Schreiben Sie ein C- Programm, das die größte von 3 über Tastatur eingegebenen, ganzen Zahlen z1, z2, z3 bestimmt und in der Variablen max speichert. Ein- und Ausgabebeteil wird nicht verlangt!

6) 10P
Das folgende syntaktisch korrekte C-Programm soll die kleinste Zahl (dreier Zahlen) berechnen und auf dem Bildschirm ausgeben.
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 min) 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 min) an, bei dem der Algorithmus falsch wird.
Geben Sie dazu genau an, wie der Fluß (Wert der Variablen in das Programm eintragen) durch das Programm verläuft (einzeichnen).

```
int main() {
    int z1, z2, z3, min;
    printf("1.Zahl eingeben:\n");
    scanf("%d", &z1);
    printf("2.Zahl eingeben:\n");
    scanf("%d", &z2);
    printf("3.Zahl eingeben:\n");
    scanf("%d", &z3);
    min = 0;
    if (z1 < z2) {
        min = z1;
    }
    if (z3 < min) {
        min = z3;
    }
    printf("Minimum = %d\n", min);
    return 0;
}
```

7) 10P
Erstellen Sie ein Struktogramm zu dem Programmausschnitt, in dem 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!

Lösung:

1)

a)

3P

Flussdiagramm, Struktogramm, Programm

b)

3P

Die Syntax definiert die äußeren Formgesetze dieser Programmiersprache (ähnlich den grammatikalischen Regeln einer natürlichen - wie z.B. der englischen- Sprache).

Die Semantik ist der Bedeutungsinhalt (ähnlich der Bedeutung der einzelnen Worte einer natürlichen - wie z.B. der italienischen - Sprache) der einzelnen Objekte einer Programmiersprache.

c)

3P

Ein Compiler ist ein Übersetzer, der einen in einer höheren Programmiersprache formulierten Text (ein sogenanntes Programm) in einen aus Maschinenbefehlen bestehenden Text (einem sogenannten Maschinenprogramm) verwandelt. Dieses kann dann vom Mikroprozessor abgearbeitet (ausgeführt) werden. Außerdem prüft er die Syntax.

2)

5P

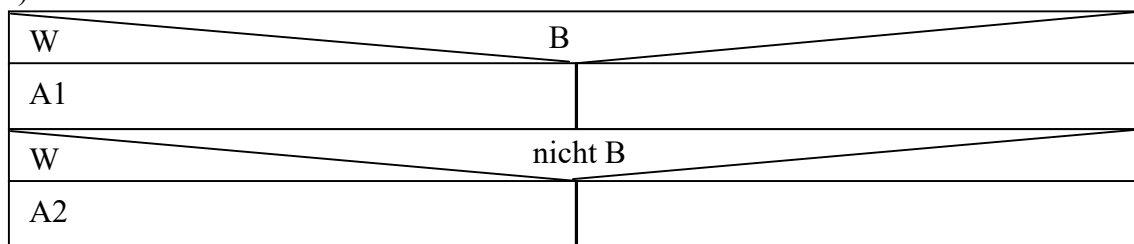
a) Die Syntax ist korrekt, da bei nur einer Anweisung im if-Teil diese nicht mit geschweiften Klammern geklamert werden muß.

b) Der Algorithmus ist nicht korrekt:

$z1 = 1000 \quad z2 = 1 \implies \min = 1000$

3)

3P



4)

5 P

$$2^{64} = 2^{60+4} = 2^{60} \cdot 2^4 = 16 \cdot 2^{60} = 16 \cdot 2^{10 \cdot 6} = 16 \cdot (2^{10})^6 \approx 16 \cdot (10^3)^6 = 16 \cdot 10^{18}$$

5)

14P

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

6)

10P

Der Algorithmus ist nicht korrekt:

$z1 = 30 \quad z2 = 20 \quad z3 = 10 \implies \min = 0$

7)

10P

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

KLAUSUR 1 Programmierpraktikum 2BK11 15.11.2019 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_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 Fehler erzeugen !!!

AUFGABEN

1)

50P

Über Tastatur sollen 2 Zahlen eingegeben werden.

Wenn nach der Eingabe aller 2 Zahlen festgestellt wird, daß alle 2 Zahlen Noten sind (also die Zahl größer gleich 1 und kleiner gleich 6 sind), muß der Mittelwert der 2 Noten auf dem Bildschirm ausgegeben werden.

Sonst muß ausgegeben werden, welche Zahlen keine Note sind.

Tipp:

Fehler bei der Eingabe können z.B. in der integer-Variablen "fehler" so verwaltet werden:

fehler = 0 : Keine Fehler bei der Eingabe (d.h. alle 2 Zahlen sind Noten)

fehler = 1 (=01) : 1. Zahl ist eine Note und 2. Zahl ist keine Note.

fehler = 10: 1. Zahl ist keine Note und 2. Zahl ist eine Note.

fehler = 11: 1. Zahl ist keine Note und 2. Zahl ist keine Note.

D.h. Eine 1 in der Zahl bedeutet welche Zahl keine Note ist !!

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.

Lösung:

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

```
int main(){
    double note1;
    double note2;
    double mittelwert;
    int zustand;

    zustand=0;
    // E I N G A B E T E I L
    printf("Bitte 1. Note eingeben\n");
    scanf("%lf", &note1);
    printf("Bitte 2. Note eingeben\n");
    scanf("%lf", &note2);

    // V E R A R B E I T U N G S T E I L
    if(note1>=1 && note1<=6 && note2>=1 && note2<=6){
        zustand=0;
        mittelwert=(note1+note2)/2.0;
    }
    else{
        if(!(note1>=1 && note1<=6) && note2>=1 && note2<=6){
            zustand=-1;
        }
        else{
            if(note1>=1 && note1<=6 && !(note2>=1 && note2<=6)){
                zustand=-2;
            }
            else{
                zustand=-11;
            }
        }
    }

    // A U S G A B E T E I L
    if(zustand==0){
        printf("Mittelwert=%f\n",mittelwert);
    }
    else{
        if(zustand==-1){
            printf("1. eingegebene Zahl ist keine Note");
        }
        else{
            if(zustand==-2){
                printf("2. eingegebene Zahl ist keine Note");
            }
            else{
                printf("Alle 2 eingegebenen Zahlen sind keine Noten");
            }
        }
    }
}
```

KLAUSUR 1 Programmierpraxis 2BKI1 Nachtermin 1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß 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

Schreiben Sie ein Programm, das drei überTastatur eingegebene Zahlen der Größe nach aufsteigend sortiert und auf dem Bildschirm ausgibt.

Bemerkungen:

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

KLAUSUR 1 Programmierpraxis 2BKI1 Nachtermin 2 Zeit: 70 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

Es soll ein Taschenrechner (er soll die 4 Grundrechenarten beherrschen) programmiert werden.

- a/A: Berechnet die Summe
- b/B: Berechnet die Differenz
- c/C: Berechnet das Produkt
- d/D: Berechnet den Quotient
- sonst: Programmende

Dieser soll auf 2 Arten (der Anwender kann dies auswählen) betrieben werden können: als Erwachsenentaschenrechner oder als Kindertaschenrechner. Der Kindertaschenrechner soll bei Eingabe einer negativen Zahl bzw. eines negativen Ergebnisses sofort eine entsprechende Meldung bringen und dann das Programm sofort beenden.

Außerdem soll der Taschenrechner (egal ob als Erwachsenentaschenrechner oder als Kindertaschenrechner genutzt) bei Division durch Null sofort eine entsprechende Meldung bringen und dann das Programm sofort (ohne Rechnung) beenden.

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.

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) 6P

Geben Sie jeweils das Struktogramm für folgende Anweisungen an:
while-Anweisung, do-while-Anweisung, for-Anweisung.

2) 3P

Simulieren Sie den folgenden Programmausschnitt durch eine do-while-Schleife:

```
s=0;
x = 1;
while (x<=2) {
    s=s-1;
    x=x+1;
}
```

3) 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");
}
```

4)

6P

Simulieren Sie die Anweisung

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

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

Für A dürfen Sie genau eine beliebige syntaktisch korrekte Anweisung und für B eine beliebige syntaktisch korrekte Bedingung wählen.

5)

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.

6)

6P

Schreiben Sie ein Programm in C, das mit Hilfe einer do-while-Schleife die folgende Summe berechnet: $10 + 11 + 12 + \dots + 9998 + 9999 + 10000$.

7)

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.

Tipp:

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

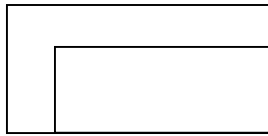

```
int main() {
    int i, j, k;
    for (i=0; i<4; i++) {
        for (j=0; j<i; j++) {
            printf(" ");
        }
        for (j=0; j<7-2*i; j++) {
            printf("#");
        }
        for (k=0; k<=i; k++) {
            printf(" ");
        }
        for (k=0; k<3; k++) {
          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

Lösungen:

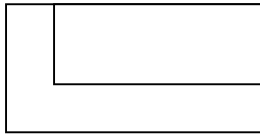
1)

6P

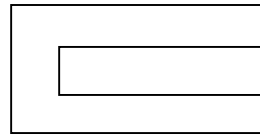
a)



b)



c)



2)

3P

```
s=0;
x = 1;
do{
    s=s-1;
    x=x+1;
}
while (x<=2) {
```

3)

12P

unendlich, 1, 0, 1, 2

4)

6P

```
bool b = true;
if(b==true){
    b=false;
}
```

wird ersetzt durch:

```
bool b = true;
while(b){
    b=false;
}
```

5)

6P

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

6)

6P

```
int main()
{
    int summe=0;
    int i=10;
    while(i<=10000){
        sum=sum+i;
        i=i+1;
    }
}
```

7)

14P

[illegible]

KLAUSUR 2 Programmierpraktikum 2BK11 13.12.2019 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.

AUFGABEN

1) 50P

Schreiben Sie ein Programm, das den Ersatzwiderstand (in Ohm) von den vom Anwender über Tastatur eingegebenen, hintereinander 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 hintereinander geschalteten Widerständen r_1, \dots, r_n gilt:

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

Bemerkung:

B1)

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

B2)

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

B3)

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:

```
#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+w;
        }
        if(w==0){
            beenden=1;
        }

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

    if(summe!=0)
        ersatz=summe;

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

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

    return 0;
}
```


KLAUSUR 2 Programmierpraxis 2BKI1 Nachtermin 1 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

Schreiben Sie ein Programm, in welchem eine über Tastatur eingegebene Dualzahl (man beginnt mit der Ziffer der niederwertigsten Stelle) in eine Dezimalzahl umgewandelt wird. Eine Zahl ungleich 0 bzw. ungleich 1 markiert das Ende der Dualzahl. (ohne EVA-Prinzip!)

Beispiel:

Eingabe: 011017

wobei die 0 links zuerst eingegeben wurde und damit die Stelligkeit $2^0 = 1$ hat.

Es wird berechnet:

$$0 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot 2^3 + 1 \cdot 2^4 = 22$$

