

KLAUSUR 1 Programmiertheorie 2BKI1 23.11.2017 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.

AUFGABEN

1) 3+4+4 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) 5 P

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

3) 14P

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!

4) 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;
}

```

5)

12P

Durch den folgenden Algorithmus (Flußdiagramm) soll die Summe

$3 + 4 + 5 + 6 + \dots + 100$ berechnet werden.

Dazu wird das Programm immer wieder an der mit <--- bezeichneten Stelle vor der Verzweigung (bei jedem Schleifendurchgang) gedanklich angehalten (Protokoll) und die Werte der Variablen i und sum protokolliert.

a) Tragen Sie dazu in der Tabelle unten die Werte von i und sum bei den ersten 5 Durchgängen ein. Bitte sum **nicht** berechnen, sondern die Summe jeweils darstellen, wie z.B. $7 + 9 + 11$.

b) Welche Werte haben i und sum, wenn das Programm das **letzte** Mal an die mit <--- bezeichnete Stelle kommt?

c) Ist das Programm semantisch korrekt (d.h. wird also durch das Programm die Summe $3 + 4 + 5 + 6 + \dots + 100$ berechnet). Bitte Begründung angeben!

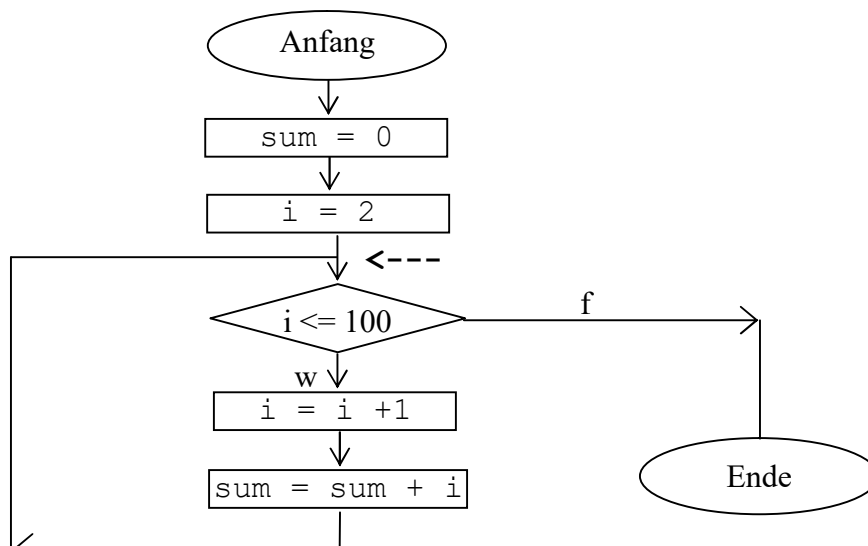


Tabelle (Protokoll):

i							
sum							

Lösung:

1)

a) 3P

Flussdiagramm, Struktogramm, Programm

b) 4P

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) 4P

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.

2) 5 P

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

3) 14P

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

4) 10P

Der Algorithmus ist nicht korrekt:

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

5) 12P

a) 5P

i	2	3	4	5	6	...	101
sum	0	0+3=3	3+4	3+4+5	3+4+5+6	...	3+...+101

b) 4P

$$i = 101 \text{ und } \text{sum} = 3 + \dots + 101$$

c) 3P

Da nach das Programm an dieser Stelle beendet wird, haben i und sum die bei b) protokollierten Werte. Das Programm ist also nicht korrekt.

KLAUSUR 1 Programmierpraktikum 2BK11 24.11.2017 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 Fehler erzeugen !!!

AUFGABEN

1) 50P

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

Dies muß mit dem EVA-Prinzip realisiert werden.

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

Erstellen Sie das dazugehörige C-Programm

Bemerkungen:

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

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

3) 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.

4) 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 r1, r2, r3, ersatz;
    int zustand=0;
    // Eingabe
    printf("Eingabe 1. Widerstand\n");
    scanf("%lf",&r1);
    if(r1<=0)
        zustand=-1;
    if(zustand==0){
        printf("Eingabe 2. Widerstand\n");
        scanf("%lf",&r2);
        if(r2<=0)
            zustand=-2;
    }
    if(zustand==0){
        printf("Eingabe 3. Widerstand\n");
        scanf("%lf",&r3);
        if(r3<=0){
            zustand=-3;
        }
    }

    // Verarbeitung
    if(zustand==0){
        ersatz=1/(1/r1+1/r2+1/r3);
    }

    // Ausgabe
    if(zustand==0){
        printf("ersatz=%f\n",ersatz);
    }
    else{
        if(zustand==-1){
            printf("1. Widerstandwert falsch");
        }
        else{
            if(zustand==-2){
                printf("2. Widerstandwert falsch");
            }
            else{
                printf("3. Widerstandwert falsch");
            }
        }
    }

    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

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) 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

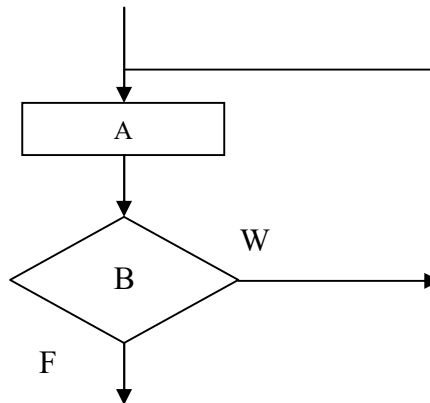
fußgesteuerte Schleife: Bedingung am Ende --> do-while-Schleife

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.

KLAUSUR 2 Programmierpraktikum 2BK11 15.12.2017 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 !!!

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

    return 0;
}
```

KLAUSUR 2 Programmiertheorie 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 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) 20P
- Erstellen Sie ein Struktogramm, das von 2 über Tastatur eingegebenen ganzen Zahlen (die jeweils als ≥ 0 vorausgesetzt werden dürfen und in den Variablen z und t gespeichert werden) berechnet, wie oft t in z enthalten ist. Dieser Wert muß in der Variablen anzahl gespeichert werden.

Der Operator / darf dabei nicht benutzt werden.

Beispiele:

t=40 ist 2 Mal in z=90 enthalten, weil $90-40-40=10$, also anzahl = 2

t=20 ist 3 Mal in z=60 enthalten, weil $60-20-20-20=0$, also anzahl = 3

t=6 ist 5 Mal in z=34 enthalten, weil $34-6-6-6-6-6=4$, also anzahl = 5

- 2) 30P
- Eine natürliche Zahl n mit $n \geq 2$ heißt Primzahl, wenn sie nur durch 1 und sich selbst teilbar ist. Beispiele für Primzahlen: 2, 3, 5, 7, 11, 13,
- Erstellen Sie ein Struktogramm, das Folgendes macht:
- Von einer über Tastatur eingegebenen ganzen Zahl n (von der vorausgesetzt werden darf, daß sie größer als 1 und ganzzahlig ist) soll bestimmt werden, ob sie eine Primzahl ist.
- Auf dem Bildschirm soll dann ausgegeben werden, ob diese Zahl eine Primzahl ist oder nicht.

Tipp: Dividieren Sie die zu überprüfende Zahl n durch 1, 2, 3, 4, ..., z und zählen wie oft es keinen Rest bei der Division gibt.

KLAUSUR 2 Programmierpraxis 2BKI1 Nachtermin 1 Zeit: 90 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß 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 Programm, in dem zwei ganze (>0) Zahlen (Zähler und Nenner) über Tastatur eingegeben werden. Außerdem soll dieses Programm

a) Zähler und Nenner maximal kürzen:

Beispiel: $18 / 24 = 3 / 4$

b) Zähler und Nenner dürfen auch negative ganze Zahlen sein ! (auch ≤ 0)

Beispiele:

$$\frac{-18}{-24} = \frac{3}{4} \quad \frac{18}{-24} = -\frac{3}{4} \quad \frac{26}{-13} = -2 \quad \frac{-26}{-13} = 2$$

KLAUSUR 3 Programmierpraktikum 2BK11 13.4.2018 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_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 !!!

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

AUFGABEN

1)

50P

Alle Felder dieses Programms sind Zeichenfelder.

Die Felder heißen string und stringCodiert.

Bei Feldlängen bitte Konstanten benutzen.

Genau beim erstmaligen Auftauchen des Zeichens 'V' im Feld string muß dieses Zeichen durch die Zeichenfolge "V3" ersetzt werden

(siehe Beispiel).

Beispiel: LEN = 10

string: Länge LEN

a	b	V	g	u	t	V	'\0'	?	?
---	---	---	---	---	---	---	------	---	---

stringCodiert: Länge LEN+1

a	b	V	3	g	u	t	V	'\0'	?
---	---	---	---	---	---	---	---	------	---

- a) 4P
Alle Zellen des Feldes `stringCodiert` müssen mit `'\0'` vorbelegt werden.
Implementieren Sie dazu den nötigen C-Quellcode.
- b) 2P
Das Feld `string` muß mit `"abVgutV"` vorbelegt werden.
Implementieren Sie dazu den nötigen C-Quellcode.
- c) 2P
Geben Sie den Inhalt des Feldes `string` auf dem Bildschirm aus.
Implementieren Sie dazu den nötigen C-Quellcode.
- d) 40P
`stringCodiert` entsteht (siehe oben) aus `string`, indem genau (nur) das erstmalige Auftauchen des Zeichens `'V'` im Feld durch die Zeichenfolge `"V3"` ersetzt wird.
Implementieren Sie dazu den nötigen C-Quellcode.
- e) 2P
Geben Sie den Inhalt des Felds `stringCodiert` auf dem Bildschirm aus.
Implementieren Sie dazu den nötigen C-Quellcode.

Bemerkungen:

B1)

Das Programm muß mit einer beliebigen Vorbelegung von `string` und `stringCodiert` funktionieren, nicht nur mit der Vorbelegung des obigen Beispiels.

B2)

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

Lösungen:

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

int main()
{
    const int LEN = 8;
    char string[LEN]="V3V";
    char stringCodiert[LEN+1];
    int i;
    int j;
    int indZeichen_V;
    int indBegin=-1;
    int indEnd;
    int indZiel;
    int stop=0;

    indBegin=0;
    indZeichen_V=-1;
    i=0;
    // Bestimme die Indizes des Teilstrings, der kopiert werden muss.
    // indBegin=-1: Zeichen 'V' kommt nicht vor
    stop=0;
    for(i=0; string[i]!='\0'; i++){
        if(string[i]=='V' && stop==0){
            indZeichen_V=i;
            indBegin=i+1;
            stop=1;
        }
        if(string[i+1]=='\0'){
            indEnd=i+1;
        }
    }

    if(indBegin==-1){
        indBegin=0;
    }

    // Bestimme Index des Ziels, an das kopiert werden soll:
    if(indZeichen_V==-1){
        indZiel=0;
    }
    else{
        indZiel=indZeichen_V+2;
    }

    // Kopiere alle Zeichen bis einschließlich 'V' nach stringCodiert
    for(i=0; i<=indZeichen_V; i++){
        stringCodiert[i]=string[i];
    }

    // Füge '3' an diese Zeichenfolge an
    if(indZeichen_V!=-1){
        stringCodiert[indZeichen_V+1]='3';
    }

    // Kopiere Rest
    j=indZiel;
    for(i=indBegin; i<=indEnd; i++){
        stringCodiert[j]=string[i];
        j++;
    }
    printf("string=\n%s\n", string);
    printf("stringCodiert=\n%s\n", stringCodiert);
    printf("indBeg=%d\n", indBegin);
    printf("indEnd=%d\n", indEnd);
    return 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_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) 10 P

Jemand geht mit 100 Euro Anfangskapital ins Spielcasino und spielt dort wie folgt Roulette:
Er setzt immer wieder 10 Euro auf die Farbe rot. Das macht er so lange, bis er entweder 110 Euro oder 0 Euro Endkapital hat, d.h. bis er insgesamt 10 Euro gewonnen oder die ganzen 100 Euro Anfangskapital verloren (verspielt) hat. Dann verlässt er das Spielcasino.
Simulieren Sie diese Strategie durch ein C-Programm (eigenes Projekt)

b) 30P

Um herauszufinden, wie groß der durchschnittliche (mittlere) Gewinn mit dieser Strategie ist, geht Herr X insgesamt $n = 100000$ Mal ins Spielcasino und kann damit seinen durchschnittliche Gewinn berechnen. Da Herr X dann bis dahin sehr alt wäre, beauftragt er Sie, ein geeignetes C-Programm zu schreiben, das diesen Sachverhalt simuliert.

Konkret:

Simulieren Sie diese bei a) eingesetzte Strategie $n = 100000$ durch ein geeignetes C-Programm und geben den durchschnittliche Gewinn dieser Strategie auf dem Bildschirm aus. (eigenes Projekt, das nicht im gleichen Projekt wie Teilaufgabe a) implementiert wird).

Bem:

Die Farbe rot - genauso wie die Farbe schwarz - erscheint jeweils mit einer Wahrscheinlichkeit von 50%.

Bemerkung:

Hier ein Demoprogramm zur Verwendung von Zufallszahlen.

```
=====

#include "stdafx.h"
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

int main(){
    int zufallszahl;
    // srand bitte nur EINMAL am ANFANG aufrufen !!!!!
    srand((unsigned)time(NULL));
    zufallszahl = rand()%6+1;
    // zufallszahl ist also einer der folgenden Zahlen:
    // 1,2,3,4,5,6
    printf("Zufallszahl=%d\n",zufallszahl);
    return(0);
}

=====
```

Lösung:

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

int main()
{
    double mittlererGewinn;
    double summeAllerGewinne=0;
    int anzahlTests=100000;
    int stop;
    double kontostand;
    int wurf;
    int i;

    // srand bitte nur EINMAL am ANFANG aufrufen !!!!!
    srand((unsigned)time(NULL));

    for(i=0;i<anzahlTests;i++){
        stop=0;
        kontostand=100;
        do{
            wurf = rand()%2;
            if(wurf==0){ // gewonnen
                kontostand = kontostand + 10;
                if(kontostand == 110){
                    stop = 1;
                }
            }
            else{
                kontostand = kontostand- 10;
                if(kontostand ==0){
                    stop = 1;
                }
            }
        }while(stop==0);
        if(kontostand==110){
            summeAllerGewinne+=10;
        }
        else{
            summeAllerGewinne-=100;
        }
    }
    mittlererGewinn = summeAllerGewinne / anzahlTests;

    printf("Mittelwert=%lf", mittlererGewinn );
    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_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) 4P

Gegeben sind folgende syntaktisch korrekte Programmteile:

a)
`i=10;
while (i<20)
 printf("Hallo Welt\n");`

b)
`i=20;
while (i<20)
 printf("Hallo Welt\n");`

Wie oft gibt der Programmteil bei a) bzw. b) die Meldung "Hallo Welt" auf dem Bildschirm aus ? Begründen Sie !

2) 6P

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

Die Eingabe einer ganzen Zahl kann man mit dem folgenden Pseudocode schreiben (wobei z eine integer-Variable bedeutet): `eingabe(z);`

3) 15P

Gegeben ist das Feld `myFeld`, in dem sich nur Zeichen (Datentyp `char`) befinden.

a) 6P

Geben Sie ein C-Programm an, das nachprüft, ob sich das Zeichen 'V' in dem Feld befindet.

b) 9P

Geben Sie ein C-Programm an, das nachprüft, ob sich die Zeichenfolge "V3" in dem Feld befindet.

4) 5P

Gegeben ist das vollständig mit Zahlen gefüllte Feld mit der Feldlänge LEN (Konstante)
`int zahlen[LEN]`

Geben Sie ein C-Programm an, das den Mittelwert dieser Zahlen berechnet.

5) 15P

Gegeben ist das **nicht** notwendig vollständig mit Zahlen gefüllte Feld der Feldlänge LEN (Konstante)

`int numbers [LEN]`

Da man irgendwo speichern muß, wie viele Zahlen sich in dem Feld befinden (es muß ja nicht vollständig mit Zahlen gefüllt worden sein), muß diese Anzahl irgendwo gespeichert werden. Hier wird die Anzahl im Element 0 des Feldes gespeichert.

Beispiel: Es befinden sich aktuell 5 Zahlen in dem Feld numbers.

5	7	-1	3	-9	4										
---	---	----	---	----	---	--	--	--	--	--	--	--	--	--	--

a) Geben Sie genau 2 Anweisungen in C an, die an das Feldende (d.h. direkt nach der Zahl 4) die Zahl 2 anfügt (auf obiges Beispiel bezogen).

b) Geben Sie genau eine Anweisung in C an, die das 1. Element (hier im Beispiel also 7) durch die Zahl 17 ersetzt (auf obiges Beispiel bezogen).

c) Geben Sie genau eine einzige Anweisung in C an, die die letzte Zahl "löscht" (auf obiges Beispiel bezogen).

d) Geben Sie genau eine einzige Anweisung in C an, die alle Zahlen "löscht". (auf obiges Beispiel bezogen).

e) Es wurden weitere Zahlen (deren Anzahl dem Programmierer unbekannt ist) in das Feld numbers geschrieben.

Geben Sie ein C-Programm an, das den Mittelwert aller Zahlen des Feldes numbers berechnet (deren Anzahl sich natürlich im 0. Element des Feldes befindet).

6) 6P

Gegeben ist das vollständig mit Zahlen gefüllte Feld mit

`ANZ_Z = 4` und `ANZ_S = 5`

`double myFeld[ANZ_Z][ANZ_S]`

Schreiben Sie ein C-Program, das in die letzte Spalte der Tabelle die Zeilensumme einträgt. Benutzen Sie dazu eine oder mehrer Schleifen (denn das Programm soll skalierbar sein, d.h. sich ohne größere Probleme auf Tabellen mit beliebiger Zeilen bzw. Spaltenzahl übertragen lassen).

Beispiel:

1	2	3	2	8
5	4	7	1	17
2	9	6	3	20
3	4	5	4	16

Lösungen:

Bemerkung: include Teil wird in allen Lösungen weggelassen!

- 1) 4P
a) Unendlich oft, weil die Bedingung immer wahr ist.
b) Null Mal, weil die Bedingung falsch ist.

2) 6P
do {
 System.out.println("ganze Zahl zwischen 1 und 5 eingeben");
 eingabe(i);
}while(!(i>=1 && i<=5));

b) 4P

Eingabe: i = 3

(!(3>=1 && 3<=5)) wird falsch und damit wird Schleife verlassen

Eingabe: i = 123

(!(123>=1 && 123<=5)) wird wahr und damit wird Schleife wiederholt.

3)
a)
int main(){
 const int LEN = 100;
 int i;
 int index=-1;
 char myFeld[LEN]="abcdeV3efghV3";

 i=0;
 while(myFeld[i]!='\0'){
 if(myFeld[i]=='V'){
 index=i;
 myFeld[i+1]='\0';
 }
 i++;
 }
 printf("V kommt an der Stelle %d vor",index);
 return 0;
}

b)
int main(){
 const int LEN = 100;
 int i;
 int index=-1;

 i=0;
 char myFeld[LEN]="abcdeV3efghV3";
 while(myFeld[i]!='\0'){
 if(myFeld[i]=='3' && i>0 && myFeld[i-1]=='V'){
 index=i;
 myFeld[i+1]='\0';
 }
 i++;
 }
 printf("V3 endet an der Stelle %d",index);
 return 0;
}


```

4)
int main(){
const int LEN = 100;
    int i;
    int zahlen[LEN];
    int summe=0;
    double mittelwert;

    for(i=0;i< LEN;i++){
        summe=summe+zahlen[i];
    }
    mittelwert=(double)summe/ LEN;
    printf("Mittelwert=%f",mittelwert);
    return 0;
}

```

5) a)
 numbers[numbers[0]+1] = 2;
 numbers[0]++;
 b) numbers[1] = 17;
 c) numbers[0] = numbers[0]-1;
 d) numbers[0] = 0;
 e)

```

int main(){
    const int LEN = 100;
    int i;
    int numbers[LEN]={5,3,4,5,6,7};
    int summe=0;
    double mittelwert;
    numbers[0]= 5;
    for(i=1;i<=numbers[0];i++){
        printf("%d",numbers[i]);
        summe=summe+numbers[i];
    }
    mittelwert=(double)summe/numbers[0];
    printf("\Mittelwert=%f",mittelwert);
    return 0;
}

```

```

6)
int main(){
    const int ANZZ=4; // Zeilenanzahl
    const int ANZS=5; // Spaltenanzahl
    int i, j;
    int myFeld[ANZZ][ANZS]={ {1,2,3,4}, {2,3,4,5}, {3,4,5,6},
                             {1,2,3,4}};

    int summe;
    for(i=0;i<ANZZ;i++){
        summe=0;
        for(j=0;j<ANZS-1;j++){
            summe=summe+myFeld[i][j];
        }
        myFeld[i][ANZS-1]=summe;
    }
    return 0;
}

```

KLAUSUR 3 Programmiertheorie 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.

AUFGABEN

1) 25P

Herr X behauptet: Wenn du das Folgende machst, kommt immer die gleiche Zahl heraus:
Schreibe eine positive dreistellige Zahl auf ein Papier. Schreibe rechts davon die gleiche Zahl nochmals auf das Papier. Dies ergibt eine sechsstellige Zahl.

Wenn man diese durch 7 dividiert, kommt immer als Rest 0 heraus.

a) Schreiben Sie ein Programm, das diese Behauptung für eine über Tastatur eingegebene Zahl überprüft, d.h. den Rest ausgibt.

b) Schreiben Sie ein C-Programm, das diese Behauptung für alle dreistellige Zahlen überprüft.

2) 25P

Entwickeln Sie ein C-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 3 Programmierpraktikum 2BKI1 22.6.2018 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

Im Fach Mathematik soll eine "Polynomfunktion 2. Grades" (" $y = ax^2 + bx + c$ ") erstellt werden. Der Mathelehrer gibt diese Aufgabe an die EDV-Abteilung weiter.

a) 20P

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

b) 20P

Implementieren Sie die Funktion fParabel(...)

c) 10P

Es soll der y-Wert einer Normalparabel an der x-Stelle 4 berechnet werden.

Schreiben Sie ein Programm, in dem ein dazu entsprechender Aufruf der Funktion fParabel(...) verwendet wird.

Geben Sie das Ergebnis auf dem Bildschirm aus.

Keine Eingabe über scanf() !!

Lösungen:

1)

a) 20P

```
/******  
/**  
/** double fParabel(double a, double b, double c, double x) **/  
/**  
/*#*****
```

Parameter:

- (i) double a : Parameter a der Parabel
- (i) double b : Parameter b der Parabel
- (i) double c : Parameter c der Parabel
- (i) double x-Wert der Parabel

Return:

ax^2+bx+c

Beschreibung:

Berechnet den y-Wert einer Parabel mit der Funktionsgleichung $y=ax^2+bx+c$ an der Stelle x.

*/

b) 20P

```
double fParabel(double a, double b, double c, double x){  
    return (a*x*x+b*x+c);  
}
```

c) 10P

```
int main(){  
    double y;  
    y= fParabel(1,0,2);  
    printf("%f",y);  
    return 0;  
}
```

KLAUSUR 4 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)

In einem aus Zeichen bestehenden Eingangsfeld (Array) soll jedes Auftreten eines bestimmten Zeichens gelöscht werden.

Diese Veränderungen dürfen aber nicht im Eingangsfeld gemacht werden.

Beispiel: In der Zeichenfolge "Mathematik" jedes Zeichen 'a' löschen.

Eingangsfeld (Array)	Ausgangsfeld (Array)
Mathematik	Mthemtik

a) 20P

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

b) 20P

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

c) 10P

In main muß ein Aufruf der Funktion "entferneZeichen(...)" 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);
}

```

Lösung:

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

```
void entferneZeichen(char stringIn[], char c, char stringOut[])
```

```
int main()
{
    int z;
    char stringIn[]="abcxyzx";
    char stringOut[]="-----";
    entferneZeichen(stringIn, 'x', stringOut);
    printf("%s",stringOut);
    scanf("%d",&z);
    return 0;
}
```

```
/**
**
** void entferneZeichen(char stringIn[], char c, char stringOut[]) **
**
**#*****
**
Parameter:
```

- (i) char stringIn[]: Eingangsfeld
- (i) char c : zu entfernendes Zeichen
- (o) char stringOut[] : Ausgangsfeld

Return:
nichts

Beschreibung:

Alle Zeichen "c", die sich in "stringIn" befinden, werden gelöscht.
Diese Aktion wird aber nicht in "stringIn" gemacht, sondern das Ergebnis nach "stringOut" kopiert.

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

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

Name, Vorname:

Hilfsmittel:

Prioritätentabelle

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

Betrachten Sie den folgenden Programmausschnitt:

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

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

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

	Adresse	Inhalt
zahl	08151	?
quad	04711	?

im Arbeitsspeicher reserviert.

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

2)

13P

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

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

3)

12P

Betrachten Sie den folgenden Programmausschnitt:

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

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

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

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

im Arbeitsspeicher reserviert.

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

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

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

`umfang = 2 * 3.14 * radius;`

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

4)

13P

Programmieren Sie anhand der folgenden Beschreibung die dazugehörige Funktion:

```

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

```

Parameter:

```

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

```

Return:

```

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

```

Beschreibung:

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

*/

Lösungen

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

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

Parameter:

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

Return:

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

Beschreibung:

Berechnet in Abhängigkeit vom Modus mod:

Summe z1+z2, wenn mod = 1

Differenz z1-z2, wenn mod = 2

Produkt z1*z2, wenn mod = 3

Quotient z1/z2, wenn mod = 4

*/

Jede fehlende Zusicherung: -2 P

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

Bemerkung zum Begriff Zusicherung:

Die Angabe beim Parameter z1:

z2!=0, wenn mod=4

und die Angabe:

mode{1;2;3;4}

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

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

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

4) 13P

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