

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 "mittlere" (d.h. zweitkleinste bzw. zweitgrößte) von 3 über Tastatur eingegebenen, ganzen Zahlen z1, z2, z3 bestimmt und in der Variablen "mittlereZahl" 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!

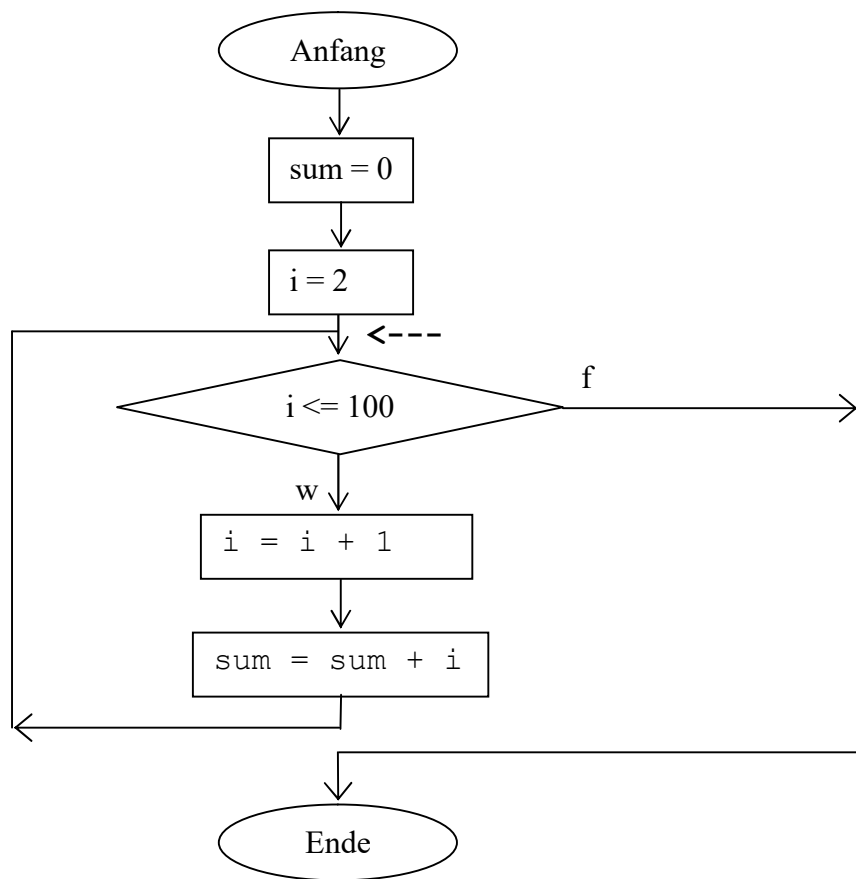


Tabelle (Protokoll):

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 && z2<=z3 || z3<=z2 && z2<=z1) {
    mittlereZahl = z2;
}
if(z2<=z1 && z1<=z3 || z3<=z1 && z1<=z2) {
    mittlereZahl = z1;
}
if(z1<=z3 && z3<=z2 || z2<=z3 && z3<=z1) {
    mittlereZahl = 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 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) 10 P

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  
}
```

2) 20 P

a) Schreiben Sie ein C-Programm (nur den Verarbeitungsteil), das von 2 Mengen A und B (die jeweils aus ganzen Zahlen bestehen) mit

$A = \{a_1, a_2\}$ mit $a_1 \neq a_2$

$B = \{b_1, b_2\}$ mit $b_1 \neq b_2$

den Durchschnitt bestimmt.

Der Durchschnitt sind genau die Zahlen, die sowohl in A als auch in B vorkommen.

b) Erstellen Sie dazu ein Testprotokoll mit 5 verschiedenen Tests.

3) 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.

b) Erstellen Sie dazu ein Testprotokoll mit 5 verschiedenen Tests.

KLAUSUR 1 Programmierpraktikum 2BK11 26.11.2013 Zeit: 45 Minuten

Gruppe A

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 Punktabzug !!!!
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.

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 Programmierpraktikum 2BK11 26.11.2013 Zeit: 45 Minuten

Gruppe B

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

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: 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) 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)

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

c) Im Ergebnis (Vereinigungsmenge bzw. Durchschnitt) 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) 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

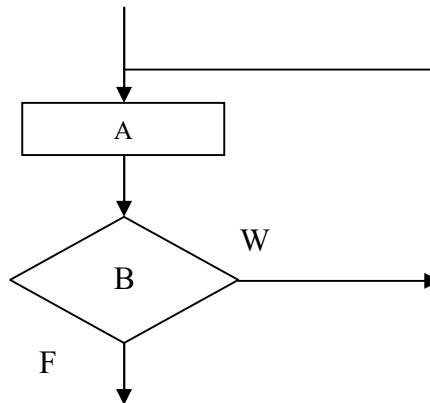
fußgesteuerte Schleife: Bedingung am Ende --> do-while-Schleife

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]

Name, Vorname:

keine

- 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.

1) 50 P

Beispiel: `anzahl = 4`

[illegible]

KLAUSUR 2 Programmiertheorie 2BKI1 Nachtermin 2 Zeit: 90 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main()) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vormane_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

AUFGABEN

- 1) 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.

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);
}
```

KLAUSUR 3 Programmierpraktikum 2BK11 20.5.2014 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)

In einem aus Zeichen bestehenden Eingangsfeld (Array) sollen zwischen 2 Stellen (jeweils einem Index) Zellen mit einem bestimmten Zeichen überschrieben werden. Diese Veränderungen dürfen aber nicht im Eingangsfeld gemacht werden. Wenn sich eine der 2 Stellen außerhalb des Feldes befinden, muß darauf entsprechend reagiert werden (entsprechender Output, keine Fehlermeldung auf dem Bildschirm)

Beispiel: In der Zeichenfolge "Mathematik" alle Zeichen zwischen der Stelle 4 und 7 mit dem Zeichen 'x' überschreiben.

Eingangsfeld (Array)	Ausgangsfeld (Array)
Mathematik	Mathxxxxik

a)

20P

Machen Sie zuerst eine Leistungsbeschreibung (Dokumentation) dieser Funktion `overwrite(...)` nach dem im Unterricht verwendeten Schema. Wo nötig (bzw. von Vorteil) den Bezeichner `const` verwenden.

b)

20P

Implementieren Sie die Funktion " `overwrite(...)` "

c)

10P

In `main` muß ein Aufruf der Funktion " `overwrite(...)` " 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.

Lösung:

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

```
int overwrite(const char stringIn[], char stringOut[], int begin,  
             int end, char zeichen);
```

```
int main(int argc, char* argv[]){  
    int erg;  
    int i;  
    char stringIn[]="abcdefg";  
    char stringOut[10];  
  
    erg=overwrite(stringIn, stringOut, 0, 6, 'x');  
  
    i=0;  
    if(erg!=0)  
        return 0;  
  
    while(stringOut[i]!='\0'){  
        printf("%c",stringOut[i]);  
        i++;  
    }  
  
    printf("\n\n");  
  
    return 0;  
}
```

```
/*  
**  
**  int overwrite(const char stringIn[], char stringOut[], int begin, **/  
**          int end, char zeichen)                                **/  
**  
**  
**#*****  
**  
*/
```

Parameter:

- (i) const char stringIn[]: Eingangsfeld
- (o) char stringOut[] : Ausgangsfeld
- (i) int begin<=end : linke Stelle
- (i) int end : rechte Stelle
- (i) char zeichen : Zeichen mit dem überschrieben wird

Return:

- 1 : begin>=len oder begin<0 oder end>=len oder end<0
oder begin>end
wobei len=Länge von stringIn
- 0 : sonst

Beschreibung:

Wenn sich "end" und "begin" innerhalb von stringIn befinden, werden von der Stelle "begin" (Zählung beginnt bei 0) bis zur Stelle "end" alle Zeichen des Strings "stringIn" mit dem Zeichen "zeichen" überschrieben und 0 zurückgeliefert, sonst wird dies nicht gemacht und -1 zurückgeliefert.

*/


```
int overwrite(const char stringIn[], char stringOut[], int begin,
              int end, char zeichen){
    int i=0;
    int len;

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

    if(begin>=len || begin<0 || end>=len || end<0)
        return(-1);

    for(i=begin; i<=end; i++){
        stringOut[i]=zeichen;
    }
    return 0;
}
```

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main()) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

AUFGABEN

1)

Im Fach Mathematik soll eine "lineare Funktion" (" $y=mx+b$ ") erstellt werden. Der Mathelehrer gibt diese Aufgabe an die EDV-Abteilung weiter.

a) 5P
Machen Sie zuerst eine Leistungsbeschreibung (Dokumentation) dieser Funktion fLinear(...) nach dem im Unterricht verwendeten Schema.

b) 4P
Implementieren Sie die Funktion fLinear(...)

c) 4P
Es soll der y-Wert einer Geraden mit der Steigung 2 und dem y-Achsenabschnitt 3 an der x-Stelle 4 berechnet werden.
Schreiben Sie ein Programm, in dem ein Aufruf der Funktion fLinear(...) verwendet wird.
Keine Eingabe über scanf() !!

2)

Die Funktion sortiere(...) sortiert ein i/o-Feld der Länge 2 von char-Werten in alphabetischer Reihenfolge.

a) 10P
Implementieren Sie diese Funktion sortiere(...).

b) 3P
Verwenden Sie einen Aufruf dieser Funktion, um das Feld zu sortieren, in dem die Zeichenfolge "za" gespeichert ist.
Keine Eingabe über scanf() !!

3)

a)

4P

Erstellen Sie die Struktur "dtPerson" (in C-Syntax), in der das Geschlecht (Datentyp char) und das Alter einer anonymen Person (ohne Namen) festgehalten werden.

b)

4P

Ein Auftraggeber verlangt: In einem C-Programm soll mit Hilfe von dynamischem Speicher genau eine 35-jährige weibliche Person angelegt werden.

Keine Eingabe über scanf() !!

c)

5P

Danach fällt dem Auftraggeber noch ein, daß er eine weitere 53-jährige männliche Person (mit Hilfe von dynamischem Speicher) anfügen will. Dies soll dadurch geschehen, daß er die schon in dem dynamischen Speicher angelegte Person in einen neu reservierten dynamischen Speicher kopiert und dann dort die neue Person anfügt. Danach wird der alte Speicher wieder freigegeben. Keine Eingabe über scanf() !!

Verwendung von dynamischem Speicher nach folgendem Schema:

```
pr = (struct dtraum *) malloc (anz*sizeof(struct dtraum));
```

4)

Betrachten Sie das folgende C-Programm (include Teil fehlt)

a)

4P

Ist dieses Programm syntaktisch korrekt?

Eventuelle Fehler begründen.

```
void belegen(int *p, int anz, int wert){
    int i;
    for(i=0; i<anz; i++){
        *(p+i)=wert+1;
    }
}
```

```
int main(){
    int zahlen[10];
    int i;
    for(i=0;i<10;i++)
        zahlen[i]=7;
    belegen(&zahlen[2], 2, 2);    // (*)
    return 0;
}
```

b)

4P

Welche Werte stehen in der Variablen "zahlen", kurz vor Abarbeitung der an der mit (*) bezeichneten Anweisung ? Bitte eintragen:

[illegible]

c)

5P

Welche Werte stehen in der Variablen "zahlen", kurz nach (aber vor Programmende)

Abarbeitung der an der mit (*) bezeichneten Anweisung? Bitte eintragen:

[illegible]

Lösungen:

1)

a) 5P

```
/******  
/**  
/** double fLinear(double m, double b, double x) **/  
/** **/  
/*#*****  
/*
```

Parameter:

- (i) double m : Steigung
- (i) double b : y-Achsenabschnitt
- (i) double x-Wert der Funktion

Return:

m*x+b

Beschreibung:

Berechnet den y-Wert einer linearen Funktion der Steigung m mit dem y-Achsenabschnitt b an der Stelle x.
*/

b) c) 8P

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

```
double fLinear(double m, double b, double x){  
    return (m*x+b);  
}
```

```
int main(){  
    double y;  
    y=fLinear(2,3,4);  
    printf("%f",y);  
    return 0;  
}
```

2)

a) 10P

```
void sortiere (int v[]){  
    int temp0, temp1;  
    temp0=v[0];  
    temp1=v[1];  
  
    if(v[0]>v[1]){  
        v[0]=temp1;  
        v[1]=temp0;  
    }  
}
```

b) 3P

```
int main(){  
    char feld[3]="az";  
    sortiere (feld);  
    return 0;  
}
```

3)

a)

4P

```
struct dtPerson{
    char geschlecht;
    int alter;
};
```

b)

4P

```
int main(){
    struct dtPerson *p, *pNeu;
    struct dtPerson person1;
    struct dtPerson person2;

    person1.geschlecht='w';
    person1.alter=35;

    person2.geschlecht='m';
    person2.alter=53;

    p = (struct dtPerson *) malloc (1*sizeof(struct dtPerson));
    p[0]=person1;
```

c)

5P

```
pNeu = (struct dtPerson *) malloc (2*sizeof(struct
                                         dtPerson));

pNeu[0]=p[0];
pNeu[1]=person2;
free(p);

printf("Geschlecht=%c Alter=%d\n",pNeu[0].geschlecht,
                                             pNeu[0].alter);
printf("Geschlecht=%c Alter=%d\n",pNeu[1].geschlecht,
                                             pNeu[1].alter);

return 0;
}
```

4)

a)

4P

Programm ist syntaktisch korrekt.

b)

4P

7	7	7	7	7	7	7	7	7	7
---	---	---	---	---	---	---	---	---	---

c)

5P

7	7	3	3	7	7	7	7	7	7
---	---	---	---	---	---	---	---	---	---

KLAUSUR 4 Programmierpraktikum 2BK11 1.7.2014 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)

In einer Zeichenfolge soll der Index des letzten Vorkommens eines Zeichens bestimmt werden. Zusätzlich soll noch die Länge der Zeichenfolge bestimmt werden.

Beispiel:

In der Zeichenfolge "Mathematik" kommt das Zeichen 't' an dem Index 7 das letzte Mal vor. Außerdem hat diese Zeichenfolge die Länge 10.

Überlegen Sie sich, was geschehen soll, wenn das Zeichen gar nicht in der Zeichenfolge vorkommt.

a) 20P

Machen Sie zuerst eine Leistungsbeschreibung (Dokumentation) der Funktion `lastIndexOf(...)` nach dem im Unterricht verwendeten Schema. Wo nötig (bzw. von Vorteil) den Bezeichner `const` verwenden.

b) 20P

Implementieren Sie die Funktion `lastIndexOf(...)`

c) 10P

Schreiben Sie ein Programm, in dem ein Aufruf der Funktion "`lastIndexOf(...)`" verwendet wird (keine Eingabe über `scanf()`, sondern Aufruf mit konkreten Parametern).

Testen Sie diese Funktion, indem die Länge der Zeichenfolge und das letztmalige Vorkommen des Zeichens auf dem Bildschirm ausgegeben werden.

```

/*****/
/**
/**  int lastIndexOf(char z, char zkette[], int *length)  **/
/**
/*#*****/
/*
Parameter:
    (i) const char z      : Zeichen
    (o) char zkette[]    : Zeichenfolge
    (i) int *length      : Länge der Zeichenfolge

Return:
    Zeichen kommt nicht in "zkette" vor: -1
    Zeichen kommt in "zkette" vor : Index des letzten Vorkommens

Beschreibung:
    Von der Zeichenfolge "zkette" wird die Länge bestimmt und
    in "length" zurückgeliefert.
    Außerdem wird der Index des letztmaligen Vorkommens des
    Zeichens "z" in der Zeichenfolge "zkette" bestimmt und
    über return zurückgeliefert.
    Falls z in "zkette" nicht vorkommt, wird -1 zurückgeliefert,
    ansonsten ein Wert >= 0
*/

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

int lastIndexOf(char z, char string[], int *length);

int main(array<System::String ^> ^args)
{
    int erg;
    int len;

    erg=lastIndexOf('x', "trixixix", &len);
    erg=lastIndexOf('x', "a", &len);
    printf("erg=%d len=%d\n",erg,len);
    scanf("%d",&len);
    return 0;
}

int lastIndexOf(char z, char zkette[], int *length){
    int indexLast=-1;
    int i=0;
    int len;

    while(zkette[i]!='\0'){
        if(zkette[i]==z){
            indexLast=i;
        }
        i++;
    }
    *length=i;
    return indexLast;
}

```