

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

Was bedeutet Priorität in der Programmiersprache C ?

Geben Sie ein Beispiel.

2) 4P

Was bedeutet Assoziativität in der Programmiersprache C ?

Geben Sie ein Beispiel.

3) 4P

Kann die Ganzzahl 123456 in einer zwei Bytes grossen integer-Zahl abgespeichert werden ?
Begründen Sie!

4) 10P

Das Volumen einer Kugel berechnet sich wie folgt:

$$V = \frac{4}{3} \cdot \pi \cdot r^3$$

Schreiben Sie das dazu passende C-Programm. Das Volumen muss in einer Variablen mit dem Datentyp float gespeichert werden. Die Konstante pi muss in einer Variablen abgespeichert werden.

5) 16P

Sie werden beauftragt ein C-Programm (Taschenrechner) zu schreiben, das abhängig von der Eingabe eines Zeichens (+, -, *, /) entweder die Summe oder die Differenz oder das Produkt oder den Quotienten zweier Zahlen liefert.

Beachten Sie bitte das EVA-Prinzip und berücksichtigen Sie, dass der Anwender auch andere Zeichen eingeben kann, als +, -, *, /.

6)

12P

Durch den folgenden Algorithmus (Flußdiagramm) soll die Summe $1 + 2 + 3 + \dots + 100$ 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 $1 + 2 + 3 + \dots + 100$ berechnet). Bitte Begründung angeben!

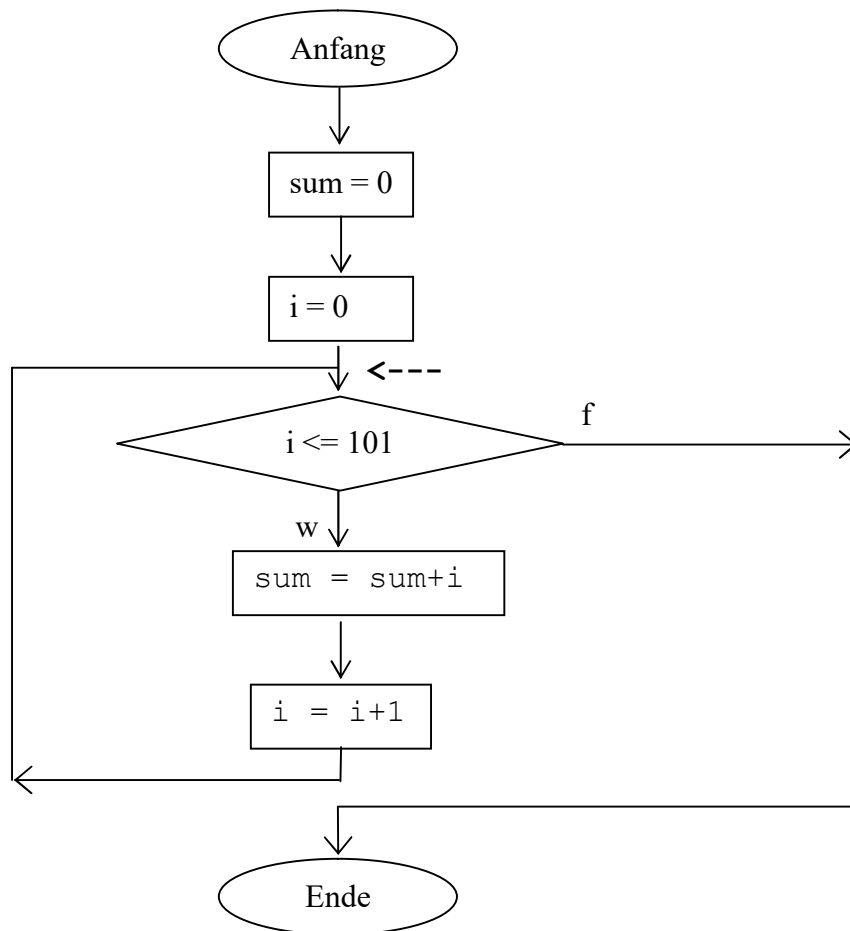


Tabelle (Protokoll):

i							
sum							

Lösungen:

1) 4P
Priorität gibt den Vorrang eines Operators in einem Ausdruck an.
Beispiel: Punkt vor Strich.

2) 4P
Assoziativität gibt an, in welcher Reihenfolge (von links nach rechts oder von rechts nach links) Operatoren mit der gleichen Priorität abgearbeitet werden.
Beispiel: * wird von links nach rechts abgearbeitet.

3) 4P
2 Bytes kann $2^{16} \approx 64000$ verschiedene Zustände speichern.
Damit können Ganzzahlen von ungefähr -32000 bis ungefähr 32000 abgespeichert werden.
Also kann 123456 in dieser zwei Bytes grossen Integer-Zahl nicht gespeichert werden.

4) 10P
#include "stdafx.h"
#include <stdio.h>

```
int main(){
    float radius;
    float volumen;
    float pi;

    // Eingabeteil
    printf("Berechnung des Volumens einer Kugel\n");
    printf("Bitte geben sie den Radius ein\n");
    scanf("%lf",&radius);
    fflush(stdin);

    // Verarbeitungsteil
    pi = (float) 3.14;
    volumen = (float)4.0/(float)3.0 *pi *radius*radius*radius;

    //Ausgabeteil
    printf("Kugelvolumen= %f\n",volumen);

    return 0;
}
```

5)

16P

```

int main(){
    double zahl1, zahl2;
    double ergebnis;
    char rechenzeichen = 's'; // schlecht

    // Eingabeteil
    printf("Bitte die 1. Zahl eingeben\n");
    scanf("%lf", &zahl1);
    fflush(stdin);

    printf("Bitte die 2. Zahl eingeben\n");
    scanf("%lf", &zahl2);
    fflush(stdin);

    printf("Bitte das Rechenzeichen eingeben\n");
    scanf("%c", &rechenzeichen);
    fflush(stdin);

    //Verarbeitungsteil
    if(rechenzeichen == '+'){
        ergebnis = zahl1 + zahl2;
        rechenzeichen = 'g'; // gut
    }

    if(rechenzeichen == '-'){
        ergebnis = zahl1 - zahl2;
        rechenzeichen = 'g'; // gut
    }

    if(rechenzeichen == '*'){
        ergebnis = zahl1 * zahl2;
        rechenzeichen = 'g'; // gut
    }

    if(rechenzeichen == '/'){
        ergebnis = zahl1 / zahl2;
        rechenzeichen = 'g'; // gut
    }

    // Ausgabeteil
    if(rechenzeichen == 'g'){
        printf("Ergebnis = %f\n", ergebnis);
    }
    else{
        printf("Bitte richtiges Rechenzeichen eingeben\n");
    }

    return 0;
}

```

6)

12 P

a)

i	0	1	2	3	4	...	102
sum	0	0	1	1+2	1+2+3	...	1+...+101

b)

i = 102 und sum = 1+...+101

c) 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 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) 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 10.12.2014 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

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

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

1)

50P

Es soll ein Taschenrechner programmiert werden.

Zuerst muss dazu ein Zeichen über Tastatur eingegeben werden:

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

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

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

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

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

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

Bemerkungen:

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

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

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

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

4) Durch 0 darf nicht dividiert werden.

Lösung

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

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

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

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

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

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

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

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

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


KLAUSUR 1 Programmierpraxis 2BKI1 Nachtermin 1 Zeit: 90 Minuten

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

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

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

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

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

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

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

Dies muß mit dem EVA-Prinzip realisiert werden.

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

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

Erstellen Sie das dazugehörige C-Programm.

Bemerkungen:

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

b)

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

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

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 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_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

AUFGABEN

1) 3P
Was ist eine Endlosschleife?

2) 8P
a) Was ist allgemein der Hauptunterschied (bzgl. der Anzahl der Durchgänge) zwischen einer while-Anweisung und einer do-while Anweisung ?
b) Geben Sie dazu jeweils (mitAnzahl der Durchgänge) ein **konkretes** Beispiel (Programmteil) an, wobei der Schleifenkörper der gleiche sein muß.

3) 9P
Wie oft wird in den folgenden Programmausschnitten jeweils die Meldung "Hallo Welt" auf dem Bildschirm ausgegeben ?

a)

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

b)

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

c)

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

4) 10P
Ein Anwender soll eine ganze Zahl zwischen 1 und 5 (je einschließlich) über Tastatur eingeben. Schreiben Sie dazu einen Programmausschnitt in C, in der der Anwender gezwungen wird. so lange eine Zahl einzugeben, bis diese Bedingung erfüllt ist. Erst dann soll im Programm die nächste Anweisung erreicht werden.

5) Gegeben ist der folgende syntaktisch korrekte Programmteil:

8P

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

Wie oft gibt dieser Programmteil die Meldung "Hallo Welt" auf dem Bildschirm aus ?
Begründen Sie !

6)

13P

Durch den folgenden Algorithmus (Flußdiagramm) soll die Summe $1 + 2 + 3 + \dots + 100$ 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.

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 $1 + 2 + 3 + \dots + 100$ berechnet). Bitte Begründung angeben!

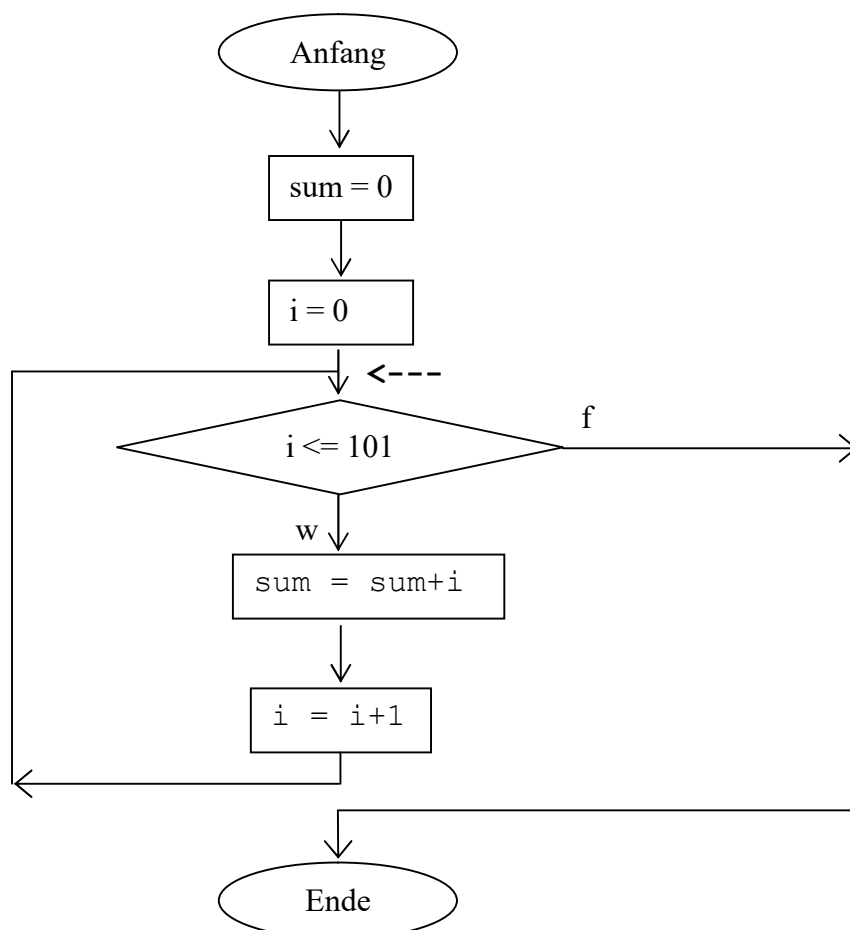


Tabelle (Protokoll):

i							
sum							

Lösungen:

1) 3P
Eine Endlosschleife ist eine Schleife, die nie verlassen wird.

2) 8P
while-Anweisung: 0 - 00
do-while Anweisung: 1 - 00

```
while (1<1) {  
    x=5;  
}
```

0 Durchgänge

```
do{  
    x=5;  
}while (1<1)
```

1 Durchgang

3) 9P
00, 1, 0

4) 10P
...
do {
 printf("ganze Zahl zwischen 1 und 5 eingeben\n");
 scanf("%d", &i);
}while(!(i>=1 && i<=5));

5) 8P
Ein Mal, weil
printf("Hallo Welt\n");
nicht zum Körper der for-Anweisung (Semikolon beachten !) gehört.

6) 13P
a)

i	0	1	2	3	4	...	102
sum	0	0	1	1+2	1+2+3	...	1+...+101

b)
i = 102 und sum = 1+...+101

c) 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 2 Programmierpraktikum 2BK11 15.1.2015 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 Programm, das den Ersatzwiderstand (in Ohm) von den vom Anwender über Tastatur eingegebenen, parallel geschalteten Widerständen berechnet und auf dem Bildschirm (in Ohm) ausgibt.

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

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

Bemerkung:

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

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

Lösung:

```
#include "stdafx.h"

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

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

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

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

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

KLAUSUR 2 Programmiertheorie 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.

AUFGABEN

1)

50P

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,

Schreiben Sie ein C-Programm, das die ersten 100 Primzahlen auf dem Bildschirm ausgibt.

EVA muß nicht eingehalten werden.

Name, Vorname:

		*						
		*	*	*				
	*	*	*	*	*			
*	*	*	*	*	*	*		

Lösungen:

```
#include "stdafx.h"

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

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

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

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)

Die Funktion split2(...) teilt eine Zeichenfolge in 2 Teilzeichenfolgen, wobei das erstmalige Vorkommen eines bestimmten Zeichens die erste Teilzeichenfolge begrenzt.

Beispiel:

Zeichenfolge: "abcasdfgfdgdf"

Begrenzer: ' f '

Die zwei Teilzeichenfolgen:

"abcasd" und "gfdgdf"

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

a) 15P

Entwerfen Sie für den "Programmierknecht" eine **Beschreibung** (Leistungsbeschreibung) der Funktion "split2" mit dem im Unterricht verwendeten Schema. (Kein Programm !!!)

Überlegen Sie sich zuerst genau, welche **Parameter** nötig sind

b) 20P

Da sich der Programmierknecht zur Zeit in einem sogenannten "Tschill-Urlaub" befindet und deshalb aktuell (und wohl auch zukünftig) nicht ansprechbar ist, muss die Funktion "split2" von Ihnen selbst implementiert werden. Implementieren Sie die Funktion "split2".

c) 5P

Schreiben Sie ein Programm mit einem Aufruf der Funktion "split2"
(keine Eingabe über scanf(), sondern Aufruf mit konkreten Parametern).

d) 10P

Was würde sich ändern, wenn man bei a) dynamischen Speicher verwenden würde.
Geben Sie die Änderungen genau an.

Lösungen:

1a)

```
/* **** */
/**
/** void split2(char str1[], char begrenzer, char str2[]) **/
/**
/*# **** */
/*
```

Parameter:

- (i/o) char str1[]: String, der in 2 Strings aufgeteilt werden soll. Außerdem wird dort auch der 1. String links des Begrenzers abgespeichert (als output)
- (i) char begrenzer: Zeichen, des 1. Vorkommen die Zerlegung angibt.
- (i) char str2[] : Der String rechts des Begrenzers.

Return:

kein

Beschreibung:

Zerlegt den String str1 an der Stelle des Begrenzers (erstmaliges Vorkommen) in die 2 Teilstrings str1 und str2
Kommt der Begrenzer nicht in str1 vor, bleibt str1 unverändert und str2 = '\0'

Beispiel:

```
str1[]: "abcfxyzffz"
begrenzer: 'f'
str2[]: "wwwwwwwwwwww"
split2(str1, begrenzer, str2);
str1[]: "abc"
begrenzer: 'f'
str2[]: "xyzffz"
*/
```

b)

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

```
void split2(char str1[], char begrenzer, char str2[]);
```

```
int main(){
    char str1[]="abdexgijkf";
    char str2[]="xxxxxxxxxxxxxxxxxx";
    char begrenzer = 'f';
    printf("str1= %s\n",str1);
    split2(str1, begrenzer, str2);
    printf("str1= %s\n",str1);
    printf("str2= %s\n",str2);

    return 0;
}
```

c)

```
void split2(char str1[], char begrenzer, char str2[]){
    int len;
    int i;
    int index;

    index=-1;
    len=0;

    // Länge der zu splittenden Zeichenkette bestimmen
    for(i=0;str1[i]!='\0';i++){
        len++;
    }

    // Index des Begrenzers bestimmen
    for(i=0; str1[i]!='\0'; i++){
        if(str1[i]==begrenzer){
            index=i;
            break;
        }
    }

    if(index==-1){
        str2[0]='\0';
    }
    else{
        str1[index]='\0';
        for(i=0;str1[index+i+1]!='\0';i++){
            str2[i]=str1[index+1+i];
        }
        str2[i]='\0';
    }
}
```

d)

Vor dem Aufruf der Funktion müßte (z.B. in main) Speicher für str1 und str2 reserviert werden (mit malloc).

Nach dem Aufruf der Funktion müßte (z.B. in main) der reservierte Speicher für str1 und str2 wieder mit free freigegeben werden.

KLAUSUR 3 Programmierpraktikum 2BK11 23.4.2015 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

Schema einer Dokumentation einer Funktion

Hinweise (unbedingt beachten):

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

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

- 1) 50P
a) 45P

Implementieren Sie die folgende Funktion und testen Sie mit einem Aufruf.

```
/* **** */
/** */
/** void split2(char str[], char begrenzer, char str1[] */
/** char str2[]) */
/** */
/*# **** */
/*
```

Parameter:

- (i) char str1[]: String, der in 2 Strings aufgeteilt werden soll.
- (i) char begrenzer: Zeichen, des 1. Vorkommen die Zerlegung angibt.
- (o) char str1 : Der String links des Begrenzers.
- (o) char str2 : Der String rechts des Begrenzers.

Return:

kein

Beschreibung:

Zerlegt den String str an der Stelle des Begrenzers
(erstmaliges Vorkommen) in die 2 Teilstrings str1 und str2
Kommt der Begrenzer "begrenzer" nicht in str vor, dann
wird str1 = str und str2 = '\0'

Beispiel:

```
str[]: "abcfxyzffz"  
begrenzer: 'f'  
str1[]: "aaaaaaaaaaaa"  
str2[]: "xxxxxxxxxxxxx"  
split2(str, begrenzer, str1, str2);  
str[]: "abcfxyzffz"  
begrenzer: 'f'  
str1[]: "abc"  
str2[]: "xyzffz"  
*/
```

b)

5P

Testen Sie die obige Funktion mit einem Aufruf.

KLAUSUR 3 Programmiertheorie 2BKI1 Nachtermin 1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

Schema einer Dokumentation einer Funktion

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)

Die Funktion atoi(...) berechnet die zu einer nur aus Ziffern bestehenden Zeichenfolge den Wert der zugehörigen Zahl.

Beispiel:

Zeichenfolge: "345"

Zugehörige Zahl: $3 \cdot 100 + 4 \cdot 10 + 5 \cdot 1$

Es darf vorausgesetzt werden, daß die Zeichenfolge nur aus Ziffern besteht und leer ist.

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

Bemerkung: Ausschnitt aus der ASCII-Tabelle:

'0' --> 48, '1' --> 49, '2' --> 50, usw.

a)

15P

Entwerfen Sie für den "Programmierknecht" eine **Beschreibung** (Leistungsbeschreibung) der Funktion "atoi" mit dem im Unterricht verwendeten Schema. (Kein Programm !!!)

Überlegen Sie sich zuerst genau, welche **Parameter** nötig sind

b)

30P

Da sich der Programmierknecht zur Zeit in einem sogenannten "Tschill-Urlaub" befindet und deshalb aktuell (und wohl auch zukünftig) nicht ansprechbar ist, muss die Funktion "atoi" von Ihnen selbst implementiert werden. Implementieren Sie die Funktion "atoi".

c)

5P

Schreiben Sie ein Programm mit einem Aufruf der Funktion "atoi"

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

Lösungen:

1a) 15P
/*****
/**
/** void atoi(char str[]) **/
/** **/
/*#*****
/*

Parameter:

- (i) char str[]!='\0' : String
und alle Zellen
des Feldes Ziffern

Return:

- (o) die als Zahl interpretierte Zeichenfolge str

Beschreibung:

berechnet die zu einer nur aus Ziffern bestehenden
Zeichenfolge den zugehörigen Zahlenwert.

Beispiel:

```
erg : ?  
erg=atoi("58");  
erg : 58  
*/
```

b) 30P

```
int atoi(char str[]){  
    int i=0;  
    int len;  
    int sum=0;  
    int faktor = 1;  
  
    for(i=0;str[i]!='\0';i++){  
    }  
    len=i;  
  
    for(i=len-1;i>=0;i--){  
        sum = sum + (str[i]-48)*faktor;  
        faktor = faktor * 10;  
    }  
  
    return sum;  
}
```

c) 5P

```
erg=atoi("345");
```


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) 16P

- a) Welchen Sinn hat die von der C-Entwicklungsumgebung "spendierte" Funktion malloc(...) ?
- b) Geben Sie ein Beispiel an (kein Program, sondern verbale Beschreibung mit Begründung), wo malloc(...) unbedingt benötigt wird?
- c) Welchen Sinn hat die von der C-Entwicklungsumgebung "spendierte" Funktion free(...) ?
- d) Geben Sie ein Beispiel an (kein Program, sondern verbale Beschreibung mit Begründung), wo free(...) unbedingt benötigt wird?

2) 6P

In einer Funktion wird

- Speicherplatz für eine lokale Variable reserviert,
- dynamisch Speicher reserviert,
- Festplattenspeicher reserviert (eine Datei angelegt und beschrieben).

Welcher Speicher steht jeweils

- a) während des Aufrufs der Funktion,
- b) nach dem Aufruf der Funktion im Hauptprogramm und
- c) nach dem Beenden des Hauptprogramms zur Verfügung?

	Während des Aufrufs	Nach dem Aufruf der Funktion	Nach Beenden des Hauptprogramms
Speicherplatz für lokale Variable			
Dynamischer Speicher			
Festplattenspeicher (Datei anlegen)			

Ein Kreuz x bedeutet : Speicher steht zur Verfügung

Kein Kreuz x bedeutet : Speicher steht nicht zur Verfügung

3)

12P

a)

Erstellen Sie die Funktion

```
void set(int *p, int anz, int wert)
```

die eine bestimmte Anzahl sich hintereinander befindlicher Elemente eines Integer-Feldes auf den Wert "wert" setzt.

b)

Gegeben ist das folgende Feld v der Länge 9

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

Rufen Sie set so auf, daß die fett markierten Elemente dieses Feldes mit 17 belegt werden.

c) Rufen Sie set(...) so auf, daß dieser Aufruf zwar syntaktisch korrekt ist, aber es zur Laufzeit Schwierigkeiten gibt.

4)

16P

a)

Erstellen Sie die Struktur "dtTier" (in C-Syntax), in der genau das Geschlecht (Datentyp: char) und die id (ID-Nummer) eines Tieres (Datentyp: int) festgehalten werden.

b)

Wie viel Speicher belegt eine Variable mit dem Datentyp "dtTier" im Arbeitsspeicher?

c)

In einem C-Programm soll mit Hilfe von dynamischem Speicher genau ein weibliches Tier mit der id = 10 und ein männliches Tier mit der id = 20 angelegt werden.

Keine Eingabe über scanf() !!

Tipp:

Verwendung von dynamischem Speicher nach folgendem Schema:

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

d)

Ein Hacker will einen Virus in das Programm einbauen, indem er ein männliches Tier mit der id = 30 in nicht reservierten Speicher ablegt.

Fügen Sie dazu die entsprechenden Programmierzeilen in c) ein (den Aufgabenteil d) als Kommentar als "Überschrift" angeben).

Keine Eingabe über scanf() !!

e)

Geben Sie den reservierten Speicher wieder frei.

Fügen Sie dazu die entsprechenden Programmierzeilen in c) ein (den Aufgabenteil e) als Kommentar als "Überschrift" angeben).

Lösungen:

- 1) 16P
- a) Damit kann man während des Programmlaufs Speicher reservieren.
- b) Wenn ein Anwender während des Programmlaufs Zahlen eingibt, deren Anzahl er bestimmen kann (z.B. über Tastatureingabe).
- c) Damit kann der reservierte Speicher wieder freigegeben werden.

d) Der Anwender reserviert während des Programmlaufs den maximal vom Betriebssystem zur Verfügung gestellten Speicher (Wetterdaten für Montag).

Diese Daten werden ausgewertet und dann nicht mehr benötigt. Wenn dieser Speicher nicht freigegeben wird, kann der Anwender im **gleichen** Pogramm keine neuen Wetterdaten für Dienstag mehr eingeben (weil er dafür vom Betriebssystem mehr keinen Speicher mehr bekommen kann).

Bem: Wenn das Programm beendet wird, wird automatisch der ganze reservierte Speicher freigegeben (dieser muß nicht mehr vorher durch free(...) freigegeben werden).

- 2) 6P
- a) während des Aufrufs der Funktion: lokale Variable, dynamischer Speicher, Datei
- b) nach den Aufruf im Hauptprogramm: dynamischer Speicher, Datei
- c) nach dem Beenden des Hauptprogramms: Datei

	Während des Aufrufs	Nach dem Aufruf der Funktion	Nach Beenden des Hauptprogramms
Speicherplatz für lokale Variable	X		
Dynamischer Speicher	X	X	
Datei	X	X	X

- 3) 12P
- a) 6P
- ```
void set(int *p, int anz, int wert){
 int i;
 for(i=0; i<anz; i++){
 *(p+i)= wert; // p[i]= wert;
 }
}
```

- b) 3P
- ```
set(&v[5], 3, 17);
// auch möglich: set(v+5, 3, 17);
```

- c) 3P
- ```
set(&v[100], 3, 17);
// auch möglich: set(v+100, 3, 17);
```

- 4) 16P
- a) 4P
- ```
struct dtTier{
    char geschlecht;
    int id;
};
```

b)
5 Byte

1P

c)

7P

```
struct dtTier{
    char geschlecht;
    int id;
};

int main(int argc, char* argv[]){
    // 1P
    struct dtTier *p;
    struct dtTier tier1;
    struct dtTier tier2;

    tier1.geschlecht='w';
    tier1.id=10;

    tier2.geschlecht='m';
    tier2.id=20;

    // 1P
    p = (struct dtTier *) malloc (2*sizeof(struct dtTier));
    // 1P
    if(p==NULL){
        return 0;
    }
    // Alternativ
    /*
        *(p+0) = tier1;
        *(p+1) = tier2;
    */
    // Alternativ
    /*
        (*p).geschlecht = 'w';          // oder: p[0].geschlecht = 'w';
        (*p).id = 10;                    // oder: p[0].id = 10;
        (*(p+1)).geschlecht = 'm';      // oder: p[1].geschlecht = 'm';
        (*(p+1)).id = 20;                // oder: p[1].id = 20;
    */

    // 1P
    p->geschlecht = 'w';
    // 1P
    p->id = 10;
    // 1P
    (p+1)->geschlecht = 'm';
    // 1P
    (p+1)->id = 20;

    printf("Geschlecht Tier1 = %c\n",p->geschlecht);
    printf("ID-Nummer Tier1 = %d\n",p->id);

    printf("Geschlecht Tier2 = %c\n", (p+1)->geschlecht);
    printf("ID-Nummer Tier2 = %d\n", (p+1)->id);

    // d)
    (p+10)->geschlecht = 'm';

    // e)
    free(p);

    return 0;
}
```

2P

2P

KLAUSUR 4 Programmierpraktikum 2BK11 25.6.2015 Zeit: 55 Minuten

Name, Vorname:

Hilfsmittel:

Schema einer Dokumentation einer Funktion

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:

- 1) Der Compiler darf keine Warnungen und Fehler erzeugen !!!
- 2) Alle Aufgaben in genau ein Programm !!!!!

AUFGABEN

1) 51P

1) 24P

In der Struktur "dtTier" wird genau das Geschlecht (Datentyp: char) und die id (ID-Nummer) eines Tieres (Datentyp: int) festgehalten.

Alle folgenden Aufgabenteile dieser Aufgabe in genau einem Programm implementieren.

a)

In einem C-Programm soll mit Hilfe von dynamischem Speicher genau ein weibliches Tier mit der id = 10 und ein männliches Tier mit der id = 20 angelegt werden.

Keine Eingabe über scanf() !!

Tipp:

Verwendung von dynamischem Speicher nach folgendem Schema:

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

b)

Ein Hacker will einen Virus in das Programm einbauen, indem er ein männliches Tier mit der id = 30 in nicht reservierten Speicher ablegt.

Fügen Sie dazu die entsprechenden Programmierzeilen in a) ein

(den Aufgabenteil b) als Kommentar als "Überschrift" angeben).

Keine Eingabe über scanf() !!

c)

Geben Sie den reservierten Speicher wieder frei.

Fügen Sie dazu die entsprechenden Programmierzeilen in a) ein

(den Aufgabenteil c) als Kommentar als "Überschrift" angeben).

2)

27P

Implementieren Sie die folgende Funktion und testen Sie diese, indem Sie die Funktion 2 Mal hintereinander aufrufen und dann die Elemente des Feldes auf dem Bildschirm ausgeben.

```

/*****
/**
/**  void vertauscheReihenfolge(int feld[], int anz)  **/
/**
/*#*****
/*

```

Parameter:

```

(i/o) int feld : Feld
                Falls kein intern benötigter Speicher
                (während der Laufzeit) reserviert werden
                kann, wird feld=NULL;
(i) int anz    : Anzahl der Elemente des Feldes

```

Return:

kein

Beschreibung:

Ordnet die Elemente des Feldes von hinten nach vorne an.
 Falls kein intern benötigter Speicher (während) der Laufzeit
 reserviert werden kann, wird feld=NULL;

Beispiel:

```

feld = 3, 4, 1, 6, 2
anz = 5
vertauscheReihenfolge(feld, 5);
feld = 2, 6, 1, 4, 3
anz = 5

```

*/

Tipp:

Reservieren Sie lokal in der Funktion Speicher für anz integer-Zahlen und kopieren in diesen reservierten Speicher das Feld "feld".

Danach kopieren Sie in umgekehrter Reihenfolge die Elemente des reservierten Speichers in das Feld "feld".

Lösungen:

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

void vertauscheReihenfolge(int feld[], int anz);
struct dtTier{                                // 4P
    char geschlecht;
    int id;
};

int main(int argc, char* argv[])
{
    // Aufgabe 1                                24P
    struct dtTier *p;                          // 2P
    struct dtTier tier1;
    struct dtTier tier2;

    tier1.geschlecht='w';
    tier1.id=10;

    tier2.geschlecht='m';
    tier2.id=20;

                                // 2P
    p = (struct dtTier *) malloc (2*sizeof(struct dtTier));
    if(p==NULL){                                // 2P
        return 0;
    }
    // Alternativen
    /*
        *(p+0) = tier1;
        *(p+1) = tier2;
        (*p).geschlecht = 'w'; oder   p[0].geschlecht = 'w';
        (*p).id = 10; oder   p[0].id = 10;
        (*(p+1)).geschlecht = 'm'; oder   p[1].geschlecht = 'm';
        (*(p+1)).id = 20; oder   p[1].id = 20;
        oder:
        p->geschlecht = 'w';                // 2P
        p->id = 10;                          // 2P
        (p+1)->geschlecht = 'm';            // 2P
        (p+1)->id = 20;                     // 2P
    */
    printf("Geschlecht Tier1 = %c\n",p->geschlecht);
    printf("Geschlecht Tier1 = %d\n",p->id);
    printf("Geschlecht Tier2 = %c\n", (p+1)->geschlecht);
    printf("Geschlecht Tier2 = %d\n", (p+1)->id);
    // Auf nicht reservierten Speicher schreiben
    (p+10)->geschlecht = 'm';                // 4P
    free(p);                                // 2P
}
```

```

// Aufgabe 2
    int i;
    int anz=5;
    int feld[5]={6,7,8,9,10};          // 2P
    vertauscheReihenfolge(feld, anz);   // 2P
    vertauscheReihenfolge(feld, anz);
    for(i=0;i<anz;i++){                // 2P
        printf("%d ", feld[i]);
    }
    return 0;
}

/*****/
/**                                     **/
/** void vertauscheReihenfolge(int feld[], int anz)          **/
/**                                     **/
/*#*****/
/*
Parameter:
    (i/o) int feld : Feld
                    Falls kein intern benötigter Speicher
                    (während der Laufzeit) reserviert werden
                    kann, wird feld=NULL;
    (i) int anz    : Anzahl der Elemente des Feldes

Return:
    kein

Beschreibung:
    Ordnet die Elemente des Feldes von hinten nach vorne an.
    Falls kein intern benötigter Speicher (während) der Laufzeit
    reserviert werden kann, wird feld=NULL;

Beispiel:
    feld = 3, 4, 1, 6, 2
    anz = 5
    vertauscheReihenfolge(feld, 5);
    feld = 2, 6, 1, 4, 3
    anz = 5
*/

void vertauscheReihenfolge(int feld[], int anz){
    int i;                                // 1P
    int *p;                               // 2P
    p = (int *) malloc(anz*sizeof(int));  // 3P
    if(p==NULL){                          // 2P
        feld=NULL;
    }
    for(i=0;i<anz;i++){                  // 5P
        p[i]=feld[i];
    }

    for(i=0;i<anz;i++){                  // 5P
        feld[i]=p[anz-i-1];
    }
    free(p);                             // 3P
}

```