

KLAUSUR 1 Programmierpraktikum 2BK11 4.11.2008 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß 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.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NICHT **ablauffähige** C-Programme werden mit 6 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 mitaufgenommen werden.
- Name der Quelldatei nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Die entsprechenden Quelldateien müssen auf Diskette abgespeichert und abgegeben werden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

1) 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 (und welcher Wert eingegeben wurde). Erstellen Sie das dazugehörige C-Programm.

Bemerkungen:

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

b) Es widerspricht nicht dem EVA-Prinzip, wenn z.B. im Eingabeteil in bestimmten Variablen bestimmte Zustände abgespeichert werden (z.B: ob der 3. Widerstandswert kleiner oder gleich 0 ist) und diese dann im Ausgabeteil abgeprüft werden.

c)

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

Lösungen:

1) Version1:

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

```
int main(){
    double R1;
    double R2;
    double R3;
    double RGesamt;
    int programmzustand=0;

    // Eingabeteil
    printf("1. Widerstand eingeben\n");
    scanf("%lf",&R1);
    if(R1>0){
        printf("2. Widerstand eingeben\n");
        scanf("%lf",&R2);
        if(R2>0){
            printf("3. Widerstand eingeben\n");
            scanf("%lf",&R3);
            if(R3>0){
                programmzustand=0;
            }
            else{
                programmzustand=-3;
            }
        }
        else{
            programmzustand=-2;
        }
    }
    else{
        programmzustand=-1;
    }

    // Verarbeitung
    if(programmzustand==0)
        RGesamt = 1/R1 + 1/R2 + 1/R3;
        RGesamt = 1 / RGesamt;

    // Ausgabeteil
    if(programmzustand==0)
        printf("Gesamtwiderstand= %f",RGesamt);
    else if(programmzustand== -1)
        printf("1. Widerstand <=0\n");
    else if(programmzustand== -2)
        printf("2. Widerstand <=0\n");
    else
        printf("3. Widerstand <=0\n");

    return 0;
}
```

Version2:

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

```
int main(){
    double R1;
    double R2;
    double R3;
    double RGesamt;
    int programzustand=0;

    // Eingabeteil
    printf("1. Widerstand eingeben\n");
    scanf("%lf",&R1);
    if(R1<=0)
        programzustand=-1;

    if(programzustand==0){
        printf("2. Widerstand eingeben\n");
        scanf("%lf",&R2);
        if(R2<=0)
            programzustand=-2;
    }

    if(programzustand==0){
        printf("3. Widerstand eingeben\n");
        scanf("%lf",&R3);
        if(R3<=0)
            programzustand=-3;
    }

    // Verarbeitung
    if(programzustand==0)
        RGesamt = 1/R1 + 1/R2 + 1/R3;

    // Ausgabeteil
    if(programzustand==0)
        printf("Gesamtwiderstand= %f",RGesamt);
    else if(programzustand===-1)
        printf("1. Widerstand <=0\n");
    else if(programzustand===-2)
        printf("2. Widerstand <=0\n");
    else
        printf("3. Widerstand <=0\n");

    return 0;
}
```

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß 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.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1)

Wie viele Zahlen kann man mit dem Datentyp int (4 Byte) abspeichern?

Geben Sie nur die Formel an (nicht das Ergebnis der Rechnung).

(5 P)

2)

Es soll über Tastatur eine **ganze** Zahl eingegeben werden.

Ist diese Zahl eine Schulnote zwischen 1 und 4 (je einschließlich) wird die Meldung "Prüfung bestanden" ausgegeben.

Ist diese Zahl die Schulnote 5 oder 6 wird die Meldung "Prüfung nicht bestanden" ausgegeben.

Sonst wird die Meldung "Diese Zahl ist keine Note" ausgegeben.

a) Realisieren Sie dies durch ein C-Programm, das genau eine switch- Anweisung (keine if-else-Anweisung, keine if-Anweisung) verwendet. (12P)

b) Realisieren Sie dies durch ein C-Programm, das genau eine if-else-if-Kette verwendet. (13P)

3)

a) Was kann man (semantisch) mit einer if-else-Anweisung, aber nicht mit einer switch-Anweisung ? (5P)

b) Was kann man (semantisch) mit einer switch-Anweisung, aber nicht mit einer if-else-Anweisung ? (gibt es so einen Fall?) (5P)

4)

Erstellen Sie ein Struktogramm zu dem Programm, das zu der in einer Prüfung erreichten Punktezahl p, (die der Lehrer über Tastatur eingibt), folgende Meldung ausgibt: (10P)

p zwischen 0 und 36 Punkte (je einschließlich): " Prüfung nicht bestanden"

p größer 36 Punkte und kleiner (oder gleich) 100 Punkte: " Prüfung bestanden"

p nicht im Bereich zwischen 0 und 100: "unzulässige Punktezahl"

Bemerkung: Die Punktezahl kann auch nicht ganzzahlig sein!

Lösung:

1) 2^{32}

2) a)

```
int main(){
    int note;
    printf("Bitte eine ganze Note eingeben\n");
    scanf("%c", &note);

    switch(note){    // int oder char
        case 1:
        case 2:
        case 3:
        case 4:
            printf("Prüfung bestanden \n");
            break;    // nicht abweisend

        case 5:
        case 6:
            printf("Prüfung nicht bestanden \n");
            break;

        default:    // falls kein case-Fall zutrifft
            printf("Dies ist keine Note \n");
            break;
    }
    return 0;
}
```

b)

```
int main(){
    int note;
    printf("Bitte eine ganze Note eingeben\n");
    scanf("%c", &note);

    if(note==1 || note==2 || note==3 || note==4){
        printf("Prüfung bestanden \n");
    }
    else if(note==5 || note==6){
        printf("Prüfung nicht bestanden \n");
    }
    else
        printf("Dies ist keine Note \n");
}
```

3)

a) switch: nur für ganzzahlige oder char Datentypen verwendbar

b) nichts

4)

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

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 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.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NICHT **ablauffähige** C-Programme werden mit 6 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 mitaufgenommen werden.
- Name der Quelldatei nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Die entsprechenden Quelldateien müssen auf Diskette abgespeichert und abgegeben werden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

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}

KLAUSUR 1 Programmiertheorie 2BKI1 Nachtermin 1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß 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.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NICHT **ablauffähige** C-Programme werden mit 6 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 mitaufgenommen werden.
- Name der Quelldatei nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Die entsprechenden Quelldateien müssen auf Diskette abgespeichert und abgegeben werden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

1) Sie werden beauftragt ein Struktogramm eines Taschenrechners zu entwerfen, das abhängig von der Eingabe eines Zeichens (+, -, *, /) entweder die Summe oder die Differenz oder das Produkt oder den Quotienten zweier Zahlen liefert.

Zuerst muss das Rechenzeichen eingegeben werden!

Wurde ein anderes Zeichen als ein Rechenzeichen (+, -, *, /) eingegeben, oder wurde bei der Division als Nenner 0 eingegeben, muß das Programm **sofort** beendet werden und auf dem Bildschirm ausgegeben werden, welcher Fehler genau gemacht wurde. Insbesondere darf dann nicht mehr eine weitere Zahl eingegeben und das Ergebnis berechnet werden.

Dies muß mit dem EVA-Prinzip realisiert werden.

Bemerkungen:

a) Es widerspricht nicht dem EVA-Prinzip, wenn z.B. im Eingabeteil in bestimmten Variablen bestimmte Zustände abgespeichert werden (z.B: ob das eingegebene Zeichen ein Rechenzeichen ist). und diese dann im Ausgabeteil abgeprüft werden.

b)

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

KLAUSUR 1 Programmierpraxis 2BKI1 Nachtermin 2 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß 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.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NICHT **ablauffähige** C-Programme werden mit 6 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 mitaufgenommen werden.
- Name der Quelldatei nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Die entsprechenden Quelldateien müssen auf Diskette abgespeichert und abgegeben werden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

2) Flächenberechnungen eines Dreiecks, Viereckes, usw.
Seitenlängen müssen positiv sein.

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß 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.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) Wandeln Sie die folgende For-Anweisung in eine Do-While-Anweisung um, indem Sie die For-Anweisung zuerst in eine While-Anweisung umwandeln und dann in eine Do-While-Anweisung. (10P)

```
for (A1; B; A3) {  
    A;  
}
```

2) (40P)

Definition:

Die Quersumme einer ganzen Zahl ist die Summe seiner Ziffern.

Beispiele:

Die Quersumme von 367 ist: $3 + 6 + 7 = 16$

Die Quersumme von 1996 ist: $1 + 9 + 9 + 6 = 25$

Die Quersumme von -13 ist: 4 (Quersumme ist immer ≥ 0)

a) Erstellen Sie ein **Struktogramm**, das von einer ganzen Zahl (Datentyp integer) die Quersumme berechnet.

Beachten Sie:

Die Quersumme ist immer ≥ 0

b) Machen Sie dazu verschiedene Tests (dokumentierte Protokolle mit Testdaten). Geben Sie jeweils an, ob die Tests erfolgreich waren oder nicht.

Lösungen

1)

```
for (A1; B; A3)
  A;
```

<==>

```
A1;
while (B) {
  A;
  A3
}
```

<==>

```
A1;
if (B) {
  do {
    A;
    A3
  } while (B)
}
```

KLAUSUR 2 Programmierpraxis 2BKI1 Nachtermin 1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß 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.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NICHT **ablauffähige** C-Programme werden mit 6 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 mitaufgenommen werden.
- Name der Quelldatei nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Die entsprechenden Quelldateien müssen auf Diskette abgespeichert und abgegeben werden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

1) Gegeben ist das Feld

`int v[100]`

Der Anwender gibt eine Zahl n zwischen 0 und 99 ein.

Dann soll das Feld zwischen dem 0-ten und dem n -ten Element (je einschließlich) der Größe nach aufsteigend sortiert und anschließend auf dem Bildschirm ausgegeben werden
Schreiben Sie das dazugehörige C-Programm.

Bemerkung:

Berechnen Sie zuerst das Minimum des 0-ten bis n -ten Elements, dann das Minimum des 1-ten bis n -ten Elements, dann das Minimum des 2-ten bis n -ten Elements, usw.

Lösungen:

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

const int len=10;
int main(){
    int v[10];
    int n, i, j, min, minIndex, temp1, temp2;

    srand((unsigned)time( NULL ));
    printf("Dies ist ein Sortierprogramm");
    printf("Bitte eine ganze Zahl zwischen 0 und %d (je einschließlich) eingeben:", len-1);
    scanf("%d", &n);

    // Feld mit Zufallszahlen füllen
    for(i=0;i<len;i++){
        // Zufallszahl zwischen 0 und len
        v[i]=rand()%1000;
    }

    // Feld ausgeben
    for(i=0;i<len;i++){
        printf("%d ", v[i]);
    }

    for(i=0;i<=n;i++){
        // Minimum zwischen i und n bestimmen
        min = v[i];
        minIndex=i;
        for(j=i;j<=n;j++){
            if(v[j]<min){
                min=v[j];
                minIndex=j;
            }
        }
        // Minimum tauschen mit v[i]
        temp1=v[i];
        temp2=v[minIndex];
        v[i]=temp2;
        v[minIndex]=temp1;
    }

    // Ausgabe
    printf("\n\n Das sortieret Feld\n");
    for(i=0;i<=n;i++){
        printf("%d ", v[i]);
    }
    return(0);
}
```

KLAUSUR 3 Programmiertheorie 2BKI1 2.4.2009 Zeit: 60 Minuten

Name, Vorname:

Hilfsmittel:

Keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß 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.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

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.

- Welchen Wert hat der Inhalt der Variablen zahl und quad an der Stelle [1] im Programm ?
- Welchen Wert hat z und zq beim Aufruf von quadrat(zahl, &quad) ?
- Was bewirkt die Anweisung `*zq = z * z` an welcher Adresse im Arbeitsspeicher ?
(konkreten Wert der Adresse und deren Inhalt angeben!)
- 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 21.4.2009 Zeit: 45 Minuten
Gruppe A Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß 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.
- 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 mitaufgenommen werden.
- Name der Quelldatei nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Die entsprechenden Quelldateien müssen auf Diskette abgespeichert und abgegeben werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

1)a) Programmieren Sie anhand der folgenden Beschreibung die dazugehörige Funktion. **40 P**

b) Im Hauptprogramm muss der Anwender eine Basis und einen Exponenten eingeben. **10 P**
können. Das Programm berechnet dann mit Hilfe der Funktion power(...) das Ergebnis und gibt es auf dem Bildschirm aus.

```
/*  
**  
**  int power(double basis, int exponent, double *erg)  **  
**  
**#*****  
*/
```

Parameter:

- (i) double basis: Grundzahl (Basis)
- (i) int exponent: Hochzahl (Exponent)
- (o) double *erg : basis ^ exponent

Return:

- 1: 0^0
- 1: sonst

Beschreibung:

Berechnet: basis hoch exponent, also basis^exponent
wobei 0^0 nicht definiert ist (return: -1).

Beispiel:

```
erg = pow(2, -4);  --> erg = 2^-4 = 1/16  
erg = pow(-2, 3); --> erg = (-2)^3 = -8  
erg = pow(0, 0);  --> erg undefiniert --> return -1  
*/
```

Name, Vorname:

Hilfsmittel:

Keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß 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.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) Es soll eine Zahlenfolge kodiert bzw. dekodiert werden.

Die Kodierung geschieht dadurch, dass zu jeder ganzen Zahl einer ganzzahligen Zahlenfolge eine bestimmte, konstante ganze Zahl dazu addiert wird.

Beispiel:

10, 7, -23, 19 ---+3---> 13, 10, -20, 22

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

a) Entwerfen Sie für den "Programmierknecht" eine **Beschreibung** (Leistungsbeschreibung) der Funktion "kodieren" mit dem im Unterricht verwendeten Schema. (Kein Programm !!!)
Überlegen Sie sich zuerst genau, welche **Parameter** nötig sind

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

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

d) Rufen Sie die Funktion "kodieren" (mit geeigneten Parametern) direkt nach dem Aufruf in c) nochmals so auf, dass die veränderte Zahlenfolge wieder ihre ursprünglichen Werte bekommt (dekodieren).

Lösungen:

1a)

```
/* **** */
/**                                     **/
/**  int kodieren (int zahlen[], int anzahl, int wert)  **/
/**                                     **/
/*# **** */
/*
```

Parameter:

(i/o) int zahlen[]: zu kodierende Zahlenenfolge
(i) int anzahl: Anzahl der Zahlen der Zahlenenfolge
(i) int wert: Zahl, die zu der Zahlenfolge dazuaddiert wird

Return:

kein

Beschreibung:

Kodiert anzahl Zahlen einer Zahlenfolge, indem ein bestimmter Wert dazuaddiert wird.

Beispiel:

zahlen[5]: {1, 2, 3, 5, 6}
anzahl: 3
wert: 100
Nach dem Aufruf:
zahlen[5]: {101, 202, 203, 5, 6}

*/

b)

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

```
void kodieren(int zahlen[], int anzahl, int wert);
```

```
int main(int argc, char* argv[]){
```

```
    int i;
    int zahlen[5]={1, 2, 3, 5, 6};
    kodieren(zahlen, 4, 100);
    for(i=0; i<4; i++){
        printf("%d ", zahlen[i]);
    }
```

```
    kodieren(zahlen, 4, -100);
    printf("\n");
    for(i=0; i<4; i++){
        printf("%d ", zahlen[i]);
    }
    return 0;
}
```

```
void kodieren(int zahlen[], int anzahl, int wert){
```

```
    int i;

    for(i=0; i<anzahl; i++){
        zahlen[i]=zahlen[i]+wert;
    }
}
```

KLAUSUR 4 Programmierpraxis 2BKI1 Nachtermin 1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß 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.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- **NICHT ablauffähige** C-Programme werden mit 6 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 mitaufgenommen werden.
- Name der Quelldatei nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Die entsprechenden Quelldateien müssen auf Diskette abgespeichert und abgegeben werden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

Erstellen Sie anhand der folgenden Beschreibung ein Struktogramm:

50P

```
/* **** */
/**
/** int berechneNullstellen (double a, double b, double c, double d,
/** double e, double anfang, double ende,
/** double nullstellen[])
/**
/* # **** */
/*
```

Parameter:

- (i) double a: Koeffizient bei x^4
- (i) double b: Koeffizient bei x^3
- (i) double c: Koeffizient bei x^2
- (i) double d: Koeffizient bei x
- (i) double e: Konstantes Glied
- (i) double anfang<ende: Intervallanfang
- (i) double ende>anfang: Intervallende
- (o) double nullstellen[]: Nullstellen des Polynoms

Return:

Anzahl der Nullstellen des Polynoms $f(x)=ax^4+bx^3+cx^2+dx+e$

Beschreibung:

Die Nullstellen (maximal 4) des Polynoms $f(x)=ax^4+bx^3+cx^2+dx+e$ werden im Intervall [anfang;ende] berechnet und in dem Feld nullstellen abgelegt. Die Nullstellen sind Näherungen.