

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) Welche Eigenschaften hat ein Algorithmus ? (4 P)

2) Was ist die ANSI Norm ? (2 P)

3) Was ist ein Literal ? (2 P)

4) Kann die Ganzzahl 29999 in einer zwei Bytes grossen Integer-Zahl abgespeichert werden ?
Begründen Sie! (4 P)

5) Warum ist folgende Anweisung nach Verwendung der scanf-Funktion sinnvoll ?
`fflush(stdin);` (2 P)

6) Schreiben Sie ein C-Programm (das beim Kompilieren keine Warnungen und Fehlermeldungen erzeugt), das die Funktion eines kleinen Taschenrechners besitzt:
Der Anwender muss zuerst zwei Zahlen (1. Zahl Datentyp float, zweite Zahl Datentyp double) eingeben, dann muss er entweder das Zeichen "+" oder irgend ein anderes Zeichen eingeben. Wenn das Zeichen "+" eingegeben wurde, muss die Summe berechnet, wenn das Zeichen "+" nicht eingegeben wurde, muss die Differenz berechnet werden. Das Ergebnis soll als **ganze** Zahl abgespeichert (Nachkommateil abschneiden) werden. EVA-Prinzip verwenden ! (16 P)

7) Was gibt der folgende, syntaktisch korrekte Programmausschnitt auf dem Bildschirm aus ?
Begründen Sie! (4 P)

```
...
int z;
z=20;
if(z=1)
    printf("Die Zahl ist 1\n");
else
    printf("Die Zahl ist ungleich 1\n");
...
```

8) Der folgende, syntaktisch korrekte Programmausschnitt soll feststellen, ob 3 über Tastatur eingegebene ganzen Zahlen gleich sind, oder nicht.

a) Geben Sie 3 **konkrete**, ganzzahlige Werte für z1, z2 und z3 an, für die das Programm eine falsche Ausgabe erzeugt. Begründen Sie.

```
...
if(z1==z2==z3)
    printf("alle Zahlen sind gleich\n");
else
    printf("nicht alle Zahlen sind gleich\n");
...
```

(4 P)

b) Ändern Sie den Programmausschnitt so ab, dass es semantisch korrekt wird (das oben Angegebene macht).

(4 P)

9) Der folgende, syntaktisch korrekte Programmausschnitt soll feststellen, ob eine über Tastatur eingegebene ganze Zahl Element der folgenden Menge ist: {2; 4; 6; 8}:

a) Geben Sie einen **konkreten** Wert für z an, für den das Programm eine falsche Ausgabe erzeugt. Begründen Sie.

Bemerkung: Es wird vorausgesetzt, dass der Benutzer auch eine ganze Zahl eingibt.

Programmausschnitt:

```
...
if(z==2||4||6||8)
    printf("Die Zahl ist entweder 2, 4, 6, oder 8");
else
    printf("Die Zahl ist nicht 2, 4, 6, oder 8");
...
```

Bemerkung:

Beachten Sie, daß der Operator == eine höhere Priorität hat als der Operator ||

(4 P)

b) Ändern Sie den Programmausschnitt so ab, dass es semantisch korrekt wird (das oben Angegebene macht).

(4 P)

Lösungen:

1) Ein Algorithmus ist ein Verfahren zur Lösung eines gegebenen Problems. Er ist eindeutig, endlich und schrittweise.

2) Die ANSI Norm ist eine Vorschrift, die festlegt, nach welchen Regeln ein C-Programm aufgebaut sein muss.

3) Literale sind Bezeichner mit einem festen Wert wie z.B. 9

4) 2 Bytes kann $2^{16} \approx 64000$ verschiedene Zustände speichern.

Damit können Ganzzahlen von ungefähr -32000 bis ungefähr 32000 abgespeichert werden.

Also kann 29999 in dieser zwei Bytes grossen Integer-Zahl gespeichert werden.

5) Mit `fflush(stdin)` werden evtl.noch im Tastaturpuffer stehende Zeichen gelöscht.

6)

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

```
int main() {
    float z1;
    double z2;
    char z;
    int erg;

    printf("Bitte 1. Zahl eingeben\n");
    scanf("%f",&z1);
    fflush(stdin);
    printf("Bitte 2. Zahl eingeben\n");
    scanf("%lf",&z2);
    fflush(stdin);
    printf("Bitte Rechenzeichen * oder / eingeben\n");
    scanf("%c",&z);
    fflush(stdin);

    if(z=='+') {
        erg=(int) (z1+z2);
    }
    else{
        erg=(int) (z1-z2);
    }

    printf("Das Ergebnis ist:= %d\n",erg);
    return 0;
}
```

7)

Ausgabe: "Die Zahl ist 1"

Der Ausdruck $z=1$ hat den Wert 1. Ein Wert ungleich 0 bedeutet, dass die Bedingung wahr ist.

8)

Eingabe: $z1=z2=z3=0$

$z1==z2==z3$, also:

$0 == 0 == 0$

$\underbrace{\hspace{1.5cm}}_1$
 $\underbrace{\hspace{2.5cm}}_0$

b)

Ersetze $z1==z2==z3$ durch $(z1==z2) \ \&\& \ (z2==z3)$

9)

Eingabe: $z1=1$

$z==2 \ || \ 4 \ || \ 6 \ || \ 8$, also:

$1==2 \ || \ 4 \ || \ 6 \ || \ 8$

$\underbrace{\hspace{1.5cm}}_0$

$\underbrace{\hspace{1.5cm}}_1$

$\underbrace{\hspace{1.5cm}}_1$

1

b)

Ersetze $z==2 \ || \ 4 \ || \ 6 \ || \ 8$ durch

$(z==2) \ || \ (z==4) \ || \ (z==6) \ || \ (z==8)$

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.
- Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

- 1) a) Ist der folgende Programmausschnitt syntaktisch korrekt ?
b) Angenommen der Programmausschnitt wäre korrekt. Was würde auf dem Bildschirm ausgegeben ? Begründen Sie!

```
...  
int z;  
if ((z=0) || 3)  
    printf("A\n");  
else  
    printf("B\n");  
...
```

- 2) Schreiben Sie ein C-Programm, das von einem eingegebenen Zeichen entscheiden soll, ob es eine Ziffer ist oder nicht.

- a) Programm soll ohne switch-Anweisung geschrieben werden.
b) Programm soll mit switch-Anweisung geschrieben werden.

- 3) Das folgende Programm soll die Fläche eines Dreiecks berechnen.

- a) Ist das folgende Programm syntaktisch korrekt ?
b) Warum ist es semantisch nicht korrekt?

```
#include "stdafx.h"  
#include <stdio.h>  
  
int main() {  
    float g,h,A;  
  
    printf("Bitte Grundseite eingeben\n");  
    scanf("%f",&g);  
    fflush(stdin);  
    printf("Bitte Hoehe eingeben\n");  
    scanf("%f",&h);  
    fflush(stdin);  
    A = 1/2*g*h;  
    printf("A= %f\n",A);  
    return 0;  
}
```

1) Stellen Sie die folgende if-else-Anweisung durch zwei einseitige Verzweigungen (if-Anweisungen) dar.

```
if(B){  
    A1;  
}  
else{  
    A2;  
}
```

KLAUSUR 1 Programmierpraktikum 2BK11 21.11.2005 Zeit: 60 Minuten
Gruppe A Name, Vorname:

Hilfsmittel:

Bücher und Skripte eigener Wahl.

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.

AUFGABEN

1) Schreiben Sie ein C-Programm, das den Ersatzwiderstand einer Parallelschaltung von 3 Widerständen berechnet (Datentyp jeweils float).

Wurde ein Widerstandswert kleiner oder gleich Null eingegeben, muß das Programm eine entsprechende Meldung auf den Bildschirm bringen (im Ausgabeteil !).

Dies muß mit dem EVA-Prinzip realisiert werden.

Bemerkung:

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

2) Schreiben Sie ein C-Programm, das das Minimum und das Maximum dreier Zahlen (Datentyp jeweils integer) berechnet und auf dem Bildschirm ausgibt.

Es dürfen nur einseitige Verzweigungen benutzt werden!

3) Schreiben Sie ein C-Programm, das die Funktion eines kleinen Taschenrechners besitzt: Der Anwender muss zuerst zwei Zahlen (Datentyp jeweils integer) eingeben, dann muss er entweder das Zeichen "*" oder irgend ein anderes Zeichen eingeben.

Wenn das Zeichen "*" eingegeben wurde, muss das Produkt berechnet, wenn das Zeichen "*" nicht eingegeben wurde, muss der Quotient berechnet werden. Das Ergebnis soll als float-Zahl abgespeichert (Nachkommateil nicht abschneiden, Datentyp float) werden.

EVA-Prinzip verwenden !

KLAUSUR 1 Programmierpraktikum 2BK11 28.11.2005 Zeit: 60 Minuten

Gruppe B Name, Vorname:

Hilfsmittel:

Bücher und Skripte eigener Wahl.

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.

AUFGABEN

Bemerkung:

In jedes Programm muß "Eingabe", "Verarbeitung" und "Ausgabe" als Kommentar mit aufgeführt werden.

1) Schreiben Sie ein C-Programm, das die Fallhöhe h (in Meter) eines frei fallenden Körpers (im luftleeren Raum) berechnet, der mit der Geschwindigkeit v (in m/s) auf dem Erdboden aufgeschlagen ist:

$$h = \frac{v^2}{2g}$$

Benutzen Sie dazu eine define-Direktive

Bemerkung:

$g = 9,81 \text{ m/s}^2$ ist der Wert für die Erdbeschleunigung.

Alle Variablen müssen den Datentyp float haben.

2) Schreiben Sie ein C-Programm, das die Funktion eines kleinen Taschenrechners besitzt: Der Anwender muss zuerst zwei Zahlen (Datentyp jeweils integer) eingeben, dann muss er entweder das Zeichen "+" oder "-" oder "*" oder "/" oder irgend ein anderes Zeichen eingeben. Wenn das Zeichen "+" eingegeben wurde, muss die Summe, wenn das Zeichen "-" eingegeben wurde, muss die Differenz, wenn das Zeichen "*" eingegeben wurde, muss das Produkt, wenn das Zeichen "/" eingegeben wurde, muss der Quotient berechnet werden. Wurde ein anderes Zeichen als "+" oder "-" oder "*" oder "/", dann muss der Anwender eine entsprechende Meldung bekommen.

3) Schreiben Sie ein C-Programm, das die zweitgrößte ("mittlere") Zahl dreier Zahlen (Datentyp jeweils integer) berechnet und auf dem Bildschirm ausgibt.

Beispiel:

9	4	3	Die zweitgrößte Zahl ist 4
7	7	4	Die zweitgrößte Zahl ist 7

entsprechende Meldung auf den Bildschirm bringen (im Ausgabeteil !).
Dies muß mit dem EVA-Prinzip realisiert werden.

Bemerkung:

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

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) Was ist allgemein der Hauptunterschied zwischen einer While-Anweisung und einer Do-Anweisung ?

Geben Sie dazu jeweils ein **konkretes** Beispiel (Programmteil) an.

2) Gegeben ist der folgende syntaktisch korrekte Programmteil:

```
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 !

3) Gegeben ist der folgende Programmteil:

```
sum = 0;
i = 1;

while(1==1){
    i = i+1;
    sum = sum +i;
-->
}
```

a)

Tragen Sie die Werte der Variablen (an der Stelle -->) i und sum bei den ersten drei Schleifendurchgängen in die Tabelle ein (ohne Berücksichtigung der Schleifenbedingung)

i					
sum					

b)

Verändern Sie den Programmausschnitt an genau 2 Stellen so, daß der Wert der Variablen sum nach Verlassen der Schleife gleich der Summe $5 + 6 + 7 + \dots + 13 + 14 + 15$ ist.

4) Gegeben ist der folgende syntaktisch korrekte Programmteil (z1, z2 sind ganze Zahlen):

```
scanf("%d",&z1);
scanf("%d",&z2);
sum = 0;
i = z1-1;
while(i<z2){
    i = i+1;
    sum = sum+i;
}
```

Was berechnet der Programmteil (Bedingungen für z1 und z2 angeben, wann was geschieht) ?

5)

Das folgende syntaktisch korrekte Programm soll die Summe aller ganzen geraden Zahlen zwei zwischen 2 vorgegebenen ganzen geraden Zahlen berechnen.

```
int main(){
    int i, sum, z1, z2, min, max;
    sum = 0;
    printf("1. Zahl eingeben\n");
    scanf("%d",&z1);
    printf("2. Zahl eingeben\n");
    scanf("%d",&z2);

    if(z1<z2){
        min=z1;
        max=z2;
    }
    else{
        min=z2;
        max=z1;
    }

    i = min;
    while(i!=max+2){
        sum = sum + i;
        i = i + 2;
    }

    printf("Summe = %d\n",sum);
    return 0;
}
```

Beispiele:

$z1 = 16, z2 = 22 \implies \text{Summe} = 16 + 18 + 20 + 22$

$z1 = 6, z2 = 8 \implies \text{Summe} = 6 + 8$

$z1 = 46, z2 = 46 \implies \text{Summe} = 46$

Geben Sie **konkret** einen Wert für z1 bzw. z2 an, (ganze Zahlen im Bereich des Datentyp int), für den das Programm durch einen Hacker zum "Absturz" gebracht werden kann.

Begründen Sie !

Lösungen:

1) Anzahl Durchgänge der do-while-Anweisung: ≥ 1

Anzahl Durchgänge der while-Anweisung: ≥ 0

```
i = 10;
do{
    printf("%d ", i);
}
while(i < 10);
```

```
i = 10;
while(i < 10){
    printf("%d ", i);
}
while(i < 10);
```

2)

Ein Mal, weil

```
printf("Hallo Welt\n");
```

nicht zum Körper der for-Anweisung (Semikolon beachten !) gehört.

3)

a)

i	2	3	4		
sum	2	2 + 3	2 + 3 + 4		

b)

```
sum = 0;
i = 4;

while(i < 15){
    i = i + 1;
    sum = sum + i;
}
```

4)

a)

Fall $z1 \leq z2$: Die Summe zwischen $z1$ und $z2$ (je einschließlich) wird berechnet

Summe = $z1 + (z1+1) + (z1+2) + (z1+3) + \dots + z2$

Beispiel: $z1 = 3, z2 = 7 \implies$ Summe = $3 + 4 + 5 + 6 + 7$

Fall $z1 > z2$: Die Summe ist 0.

5)

Beispiel: $z1 = 5, z2 = 8 \implies$ Endlos-Schleife, weil 5 (ungerade Zahl) bei jedem Durchgang um 2 erhöht wird (und deshalb ungerade bleibt) und nie 8 werden kann, weil 8 eine gerade Zahl ist.

KLAUSUR 2 Programmiertheorie 2BKI1 Nachtermin 1 Zeit: 60 Minuten

Name, Vorname:

Hilfsmittel:

Keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß 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) a) Berechnen Sie durch ein C-Programm, durch die Verwendung einer while-Anweisung und eines hochzählenden ($i = i + 1$) Schleifenzählers die Summe von:

$$3^2 + 4^2 + 5^2 + \dots 98^2 + 99^2 + 100^2$$

b) Machen Sie das gleiche durch die Verwendung eines runterzählenden ($i = i - 1$) Schleifenzählers (nur den Teil angeben, der sich ändert)

c) Machen Sie das gleiche durch die Verwendung einer do-while Schleife. (nur den Teil angeben, der sich ändert)

d) Berechnen Sie durch ein C-Programm die Summe (wobei $z1^2$ und $z2^2$ noch zur Summe gehören) von:

$$z1^2 + (z1+1)^2 + (z1+2)^2 + \dots + (z2-2)^2 + (z2-1)^2 + z2^2,$$

wobei $z1$ und $z2$ durch den Anwender eingegeben werden und ganze Zahlen sein müssen.

Das Programm muss nur funktionieren für $z1 \leq z2$

Der Eingabeteil muss nicht implementiert (programmiert) werden.

KLAUSUR 2 Programmierpraktikum 2BKI1 13.02.2006 Zeit: 60 Minuten
Gruppe B *Name, Vorname:*

Hilfsmittel:

Bücher und Skripte eigener Wahl.

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.

AUFGABEN

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 bestimmt, ob eine ganze Zahl n ($n \geq 2$) eine Primzahl ist:

Sie geben über Tastatur eine ganze Zahl n ($n \geq 2$) ein. Auf dem Bildschirm soll dann ausgegeben werden, ob diese Zahl eine Primzahl ist oder nicht.

KLAUSUR 2 Programmierpraktikum 2BK11 17.02.2006 Zeit: 60 Minuten

Gruppe B Name, Vorname:

Hilfsmittel:

Bücher und Skripte eigener Wahl.

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.

AUFGABEN

Schreiben Sie ein Programm, das die Potenz a^n einer reellen Zahl a bestimmt.
 n muß eine **ganze** Zahl sein.

Definition der Potenz:

$$a^n = \underbrace{a \cdot a \cdot \dots \cdot a}_{n \text{ - mal}}, \text{ wenn } n > 0$$

$$a^n = 1, \text{ wenn } n = 0$$

$$a^{-n} = \frac{1}{a^n}, \text{ wenn } n < 0$$

Beispiele:

$$2^4 = 2 \cdot 2 \cdot 2 \cdot 2 = 16$$

$$5^0 = 1$$

$$3^{-2} = \frac{1}{3^2} = \frac{1}{9}$$

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) Betrachten Sie das folgende, **syntaktisch korrekte** Programm:

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

int main() {
    int istPrimzahl=1;
    int i=3;
    int istNichtTeiler, zahl;

    printf("Bitte Zahl >= 2 eingeben\n");
    scanf("%d",&zahl);
    fflush(stdin);

    if(zahl%2==0) {
        istPrimzahl=0;
    }
    else{
        while(i<zahl){
            istNichtTeiler=zahl%i;
            if(istNichtTeiler>0) {
                istPrimzahl=1;
            }
            else{
                istPrimzahl=0;
            }
            i=i+1;
        }
    }
    if(istPrimzahl==0)
        printf("%d ist keine Primzahl\n", zahl);
    else
        printf("%d ist eine Primzahl\n", zahl);
    return 0;
}
```


Das obige, syntaktisch korrekte Programm soll von einer Zahl ≥ 2 bestimmen, ob sie Primzahl ist. Es wird **vorausgesetzt**, dass nur ganze Zahlen ≥ 2 eingegeben werden. Es kann 0, 1, 2, 3, ... unendlich viele, ganze Zahlen ≥ 2 geben, bei denen das Programm semantisch falsch wird, d.h. auf dem Bildschirm falsche Aussagen ausgegeben werden.

Falls es endlich viele, ganze Zahlen ≥ 2 geben sollte, bei denen das Programm semantisch falsch wird, müssen alle Zahlen angegeben werden.

Falls es unendlich viele, ganze Zahlen ≥ 2 geben sollte, bei denen das Programm semantisch falsch wird, müssen konkret 2 Zahlen angegeben werden.

Außerdem muss **begründet** werden, warum semantische Fehler bei unendlich vielen Zahlen auftreten. Zusätzlich müssen alle Zahlen bei denen das Programm semantisch falsch wird, in einer beschreibenden Form (siehe Mengenlehre) oder mit Hilfe der deutschen Sprache angegeben werden

Lösung:

Es gibt unendlich viele Zahlen, bei denen das Programm semantisch falsch wird.

Zum Beispiel wird das Programm bei Eingabe von 2 oder 9 semantisch falsch.

Die Menge aller Zahlen ≥ 2 , für die das Programm falsch wird werde mit X bezeichnet.

Dann gilt:

$X = \{2\} \cup$ Menge aller ungeraden Nichtprimzahlen, die größer oder gleich 9 sind.

KLAUSUR 3 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.
- Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1)

Ein Glied $a(i)$ einer Folge berechnet sich aus der Summe der beiden vorhergehenden Folgenglieder $a(i-1)$ und $a(i-2)$: $a(i) = a(i-1) + a(i-2)$

Die beiden Anfangsglieder der Folge sind z.B.

$$a(1) = 5$$

$$a(2) = 3$$

Das ergibt die Folge:

$$a(1) = 5$$

$$a(2) = 3$$

$$a(3) = a(2) + a(1) = 3 + 5 = 8$$

$$a(4) = a(3) + a(2) = 8 + 3 = 11$$

$$a(5) = a(4) + a(3) = 11 + 8 = 19$$

...

also:

5, 3, 8, 11, 19, ...

Erstellen Sie ein Struktogramm, das bei Eingabe einer Zahl $n \geq 1$ das Folgenglied $a(n)$ berechnet.

KLAUSUR 3 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.
- Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1)

Ein Glied $a(i)$ einer Folge berechnet sich aus der Summe der beiden vorhergehenden Folgenglieder $a(i-1)$ und $a(i-2)$: $a(i) = a(i-1) + a(i-2)$

Die beiden Anfangsglieder der Folge sind z.B.

$$a(1) = 5$$

$$a(2) = 3$$

Das ergibt die Folge:

$$a(1) = 5$$

$$a(2) = 3$$

$$a(3) = a(2) + a(1) = 3 + 5 = 8$$

$$a(4) = a(3) + a(2) = 8 + 3 = 11$$

$$a(5) = a(4) + a(3) = 11 + 8 = 19$$

...

also:

5, 3, 8, 11, 19, ...

Erstellen Sie ein C - Programm, das bei Eingabe einer Zahl $n \geq 1$ das Folgenglied $a(n)$ berechnet.

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, positive, 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) dieser Funktion 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 von Ihnen selbst implementiert werden. Implementieren Sie diese Funktion.

c) Schreiben Sie ein Programm mit einem Aufruf dieser Funktion.