

KLAUSUR 1 Programmierpraktikum 2BK11 8.11.2004 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
- 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.

Bei Eingabe eines Widerstandswerts kleiner oder gleich Null, muß das Programm sofort beendet werden (ohne den Ersatzwiderstand zu berechnen).

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.

Bemerkung:

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.

KLAUSUR 1 Programmierpraktikum 2BK11 16.11.2004 Zeit: 60 Min.
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
- 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 die Schnittmenge und 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.

Es soll berechnet und dann ausgegeben werden:

$A \cap B$

$A \cup B$

Name, Vorname:

Hilfsmittel:

Ein DIN A4-Blatt selbst verfasste, handschriftliche Notizen

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 gibt der folgende Programmausschnitt auf dem Bildschirm aus ? Begründen Sie!

```
int x=10;
int a=0;
int b=0;
int c=0;
if(x=2)
    printf("Ausgabe1\n");
if(a==b==c)
    printf("Ausgabe2\n");
if(123)
    printf("Ausgabe3\n");
if(5-3/4==5+3/4)
    printf("Ausgabe4\n");
if(x=2&&8)
    printf("Ausgabe5\n");
```

2) Warum erscheint beim Kompilieren des folgenden syntaktisch korrekten Programmausschnitts eine Warnung ? Begründen Sie!

```
...
float f;
f = 3.5 + 4 * 2.5;
```

3) Ersetzen Sie folgenden syntaktisch korrekten Programmausschnitt durch eine verschachtelte ein- oder zweiseitige Verzweigung:

Programmausschnitt:

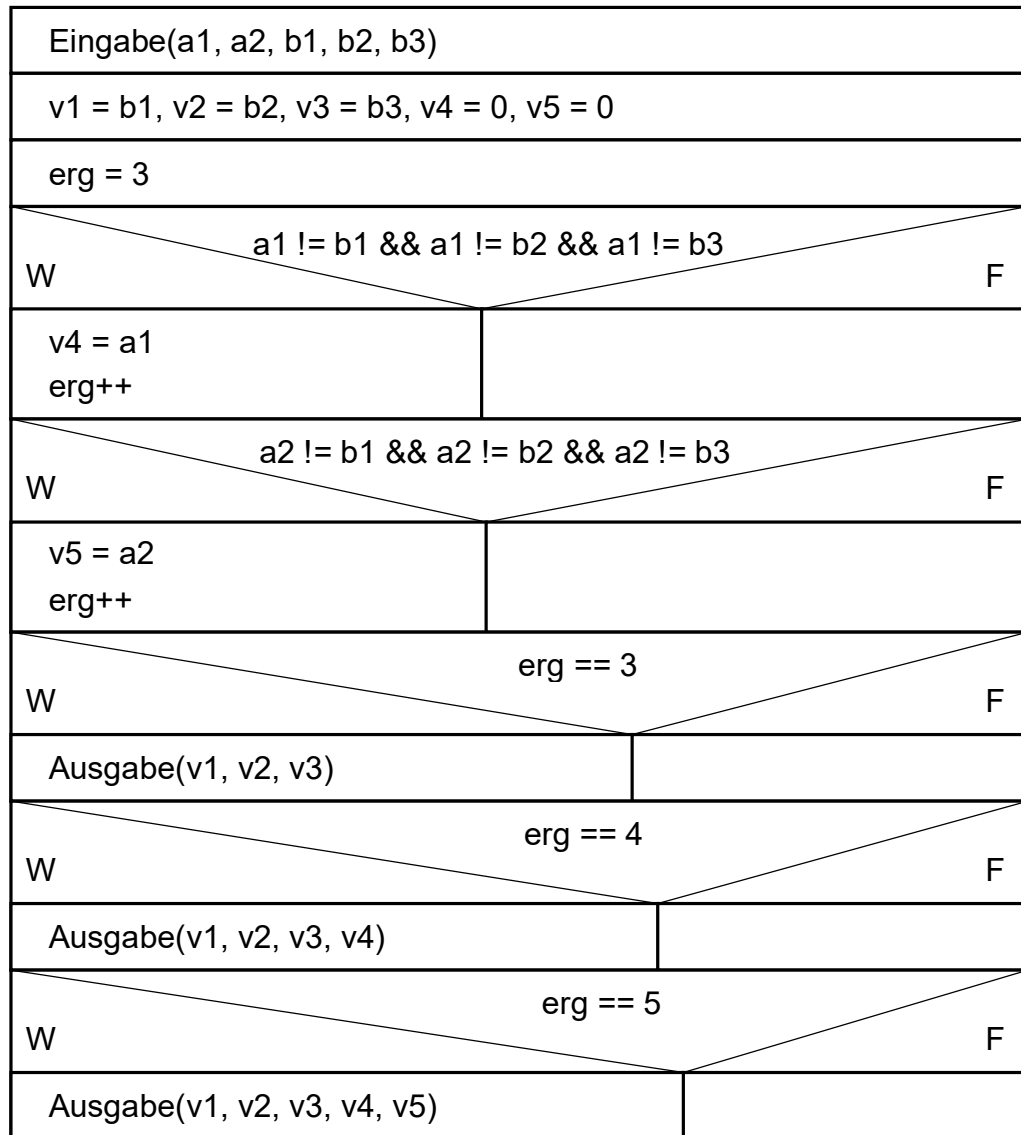
```
...
if(a==b && b==c)
    x=1;
else
    x=2;
```

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:

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"

5) Der durch das folgende Struktogramm beschriebene Algorithmus soll die Vereinigungsmenge der Zahlenmengen A und B mit $A = \{a1, a2\}$ und der $B = \{b1, b2, b3\}$ berechnen.

Begründen Sie, ob der Algorithmus korrekt ist. Falls er nicht korrekt ist, muß dazu ein konkretes Gegenbeispiel angegeben werden (mit konkreten Werten für a1, a2, b1, b2, b3)



Lösungen:

1)

a) Ausgabe1: Der Wert des Ausdrucks (Zuweisungsausdrucks) $x=2$ ist 2. Der Wert 2 wird als wahr interpretiert.

b) keine Ausgabe2: Der Wert des Ausdrucks $b==c$ ist 1, also wahr, da b und c jeweils 0 ist und damit b gleich groß wie c ist. Der Wert des Ausdrucks $a==(b==c)$, also konkret $a==1$, ist 0, also falsch, da a gleich 0 ist. Der Wert 0 wird als falsch interpretiert.

c) Ausgabe3: 123 wird als wahr interpretiert.

d) Ausgabe4: $5-3/4$ ist 5, da $3/4$ Null ist. $5+3/4$ ist 5, da $3/4$ Null ist. Also ist $5-3/4$ gleich $5+3/4$. Also hat der Ausdruck $5-3/4==5+3/4$ den Wert 1 und wird als wahr interpretiert.

e) Ausgabe5: 2 und 8 wird jeweils als wahr interpretiert. Deshalb ist $2 \ \&\& \ 8$ wahr und hat den Wert 1. Der Wert des Zuweisungsausdrucks $x=2\ \&\& \ 8$ ist deshalb 1 und wird als wahr interpretiert.

2) 4 ist integer, 2.5 ist double. Also wird 4 in double umgewandelt. Das Ergebnis 10 ist damit double. 3.5 ist double und damit ist $3.5 * 10$ auch double. Da double in float abgespeichert wird, wird die double Zahl $3.5 * 10$ in float umgewandelt. Dabei kann ein Datenverlust entstehen. Deshalb gibt der Compiler eine Warnung aus.

3)

```
if (a==b)
    if (b==c)
        x=1;
    else
        x=2;
else
    x=2;
```

4)

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

5)

a1 = 3,
a2 = 100,
b1 = 3,
b2 = 4,
b3 = 5

Name, Vorname:

Hilfsmittel:

Ein DIN A4-Blatt selbst verfasste, handschriftliche Notizen

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) Geben Sie das Struktogramm eines Programms (mit einer while Anweisung realisiert) an, das die Zahlen 10, 11, ..., 19, 20 auf dem Bildschirm ausgibt.
b) Geben Sie das Struktogramm eines Programms (mit einer do - while Anweisung realisiert) an, das die Zahlen 10, 11, ..., 19, 20 auf dem Bildschirm ausgibt.
c) Geben Sie das Struktogramm eines Programms (mit einer for Anweisung realisiert) an, das die Zahlen 10, 11, ..., 19, 20 auf dem Bildschirm ausgibt.

2) In den folgenden Programmausschnitten gibt der Anwender für i einen Wert ein.

- a) Für welche Werte von i bringen die Programmausschnitte die gleiche Anzahl von Meldungen auf den Bildschirm ?
b) Für welche Werte von i bringen die Programmausschnitte nicht die gleiche Anzahl von Meldungen auf den Bildschirm ?

```
scanf ("%d", &i) ;  
while (i<=3) {  
    i = i+1;  
    printf("Hallo Welt\n");  
}
```

```
scanf ("%d", &i) ;  
do {  
    i = i+1;  
    printf("Hallo Welt\n");  
} while (i<=3) ;
```

3) Gegeben ist der folgende Programmteil:

```
void main() {
    int i, sum;
    i=1;
    sum=1;
    do{
        i=i+1;
        ---->
        sum=sum+i;
    } while(i<=10);

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

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 das Programm an genau 1 Stelle so, daß der Wert der Variablen sum nach Verlassen der Schleife gleich der Summe $1 + 2 + 3 + \dots + 99 + 100$ ist.

4) Wandeln Sie im folgenden Programmteil die do ... while Anweisung in eine while Anweisung um, so daß der dadurch entstehende neue Programmteil genau das gleiche macht, wie der alte Programmteil.

```
scanf ("%d", &i);
do{
    i = i+1;
    printf("Hallo Welt\n");
} while(i<=30);
```

5) a) Was ist brute force ?

b) Wo wird brute force angewendet ?

c) In einem sogenannten magischen 4×4 Quadrat müssen die Zahlen von 0 bis 15 so angeordnet werden, daß jeweils die Zeilensummen, die Spaltensummen und die Diagonalsummen gleich sind und in jeder Zelle des Quadrats eine andere Zahl steht. Dies soll durch ein Programm gelöst werden, das brute force benutzt.

Wie viele Möglichkeiten müssen in diesem Programm untersucht werden ?

d) Geben Sie eine zeitliche Abschätzung für die Rechenzeit dieses Programms, unter der Voraussetzung, daß ein im Unterricht mit brute force gelöstes Zahlenrätsel mit 10 verschiedenen Zahlen z.B. eine Rechenzeit von 1 Minute benötigt.

Lösungen:

1)

i=10
while (i<=20)
Ausgabe (i)
i=i+1

i=10
Ausgabe (i)
i=i+1
while (i<=20)

for (i=10;i<=20;i++)
Ausgabe (i)

2)

i <= 3: gleiche Anzahl von Meldungen

i > 3 : verschiedene Anzahl von Meldungen

3)

a)

i	2	3	4		101
sum	1	1+2	1+2+3		1+...100

b) statt i <= 10 muß man i <= 99 schreiben.

4)

```
i = i+1;
printf("Hallo Welt\n");
```

```
while (i<=30) {
    i = i+1;
    printf("Hallo Welt\n");
}
```

5)

a) mit "roher Gewalt" werden sämtliche Möglichkeiten durchgerechnet.

b) Entschlüsseln von Passwörtern.

c) 16^{16}

d) $16^{16} \geq 10^{16} = 10^6 \cdot 10^{10}$

Das Programm braucht also mindestens 1 Million Minuten \approx 694 Tage Rechenzeit.

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 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) Es soll die Summe der Zahlen zwischen 80 und 90, also $80 + 81 + \dots + 90$ berechnet werden.

Dies kann durch eine while-Anweisung, eine do-while-Anweisung oder eine for-Anweisung realisiert werden.

Erstellen Sie für jede der 3 Möglichkeiten jeweils ein Struktogramm.

2)

Es sollen (je einschließlich) alle ganzen Zahlen zwischen der Zahl g (die der Anwender über Tastatur eingibt) und der Zahl 50 auf dem Bildschirm ausgegeben werden.

Es wird vorausgesetzt (und muß nicht mehr im Programm überprüft werden), daß der Anwender eine ganze Zahl eingibt. Diese ganze Zahl kann auch größer als 50 sein. Dann wird nichts auf dem Bildschirm ausgegeben.

a) Realisieren Sie dies durch ein Struktogramm, das eine do-while-Anweisung enthält.

b) Realisieren Sie dies durch ein Struktogramm, das eine for-Anweisung enthält.

Beispiel ($z = 10$):

10, 12, 14, 16,, 49, 50

Beispiel ($z = 50$):

50

3)

Wandeln Sie die folgende do-while Anweisung in eine for-Anweisung um:

```
do{  
    A;  
}while B
```

4)

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,

Erstellen Sie ein Struktogramm eines Programms, 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 3 Programmiertheorie 2BKI1 24.2.2005 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

Ein DIN A4-Blatt selbst verfasste, handschriftliche Notizen

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)

In einem C-Programm wurde folgende Variable deklariert:

```
char w[4];
```

In dem Programm stehen genau die folgenden Anweisungen:

```
w[0] = 'w';  
w[1] = 'i';  
w[2] = 'r';
```

Es soll ein Programm(ausschnitt) geschrieben werden, in dem der Inhalt der Variablen w (also die Zeichenkette "wir") ausgegeben werden soll.

- a) Realisieren Sie dies durch die Verwendung der Formatierung "%s" in printf
- b) Realisieren Sie dies durch die Verwendung der Formatierung "%c" in printf

2) Wieviel Speicherplatz (in Bytes) benötigen die Variablen v1, v2, v3 des folgenden Programmausschnitts ?

```
int v1[10];  
int v2[10][10];  
int v3[10][10][10];
```

3) Die Zeilensummen (einer Zahlentabelle) sollen jeweils in der letzten Spalte einer Zeile und die Spaltensummen sollen jeweils in der letzten Zeile einer Spalte abgespeichert werden. Außerdem soll auch die Gesamtsumme an einer geeigneten Stelle der Tabelle abgespeichert werden.

Realisieren Sie das durch ein zweidimensionales Feld ff mit Hilfe eines Struktogramms.

Zeilenanzahl: ANZZ,

Spaltenanzahl: ANZS

(Als Zeilenanzahl und Spaltenanzahl keine feste Zahl wählen!!)

4) Ein Text ist in einem zweidimensionalen Feld `ff` (Quadrat) abgespeichert. Um ihn vor neugierigen Blicken zu schützen wird er so codiert, dass die 1. Zeile mit der 1. Spalte, die 2. Zeile mit der 2. Spalte, usw. vertauscht werden.

Realisieren Sie das mit Hilfe eines Struktogramms.

Zeilenanzahl: `LEN`

Spaltenanzahl: `LEN`

(Als Zeilenanzahl und Spaltenanzahl keine feste Zahl wählen!!)

Tipp: Überlegen Sie sich, wo das Element `ff[i][j]` nach seiner Vertauschung steht!

5) Gegeben ist ein zweidimensionales Feld `ff` mit der Zeilenanzahl `ANZZ` und der Spaltenanzahl `ANZS`.

Die 1. Zeile dieses zweidimensionalen Feld `ff` wird in das eindimensionale Feld `f` (am Anfang des Feldes `f` beginnend) kopiert.

Gleich dahinter wird die 2. Zeile des zweidimensionalen Feldes `ff` angefügt.

Gleich dahinter wird die 3. Zeile des zweidimensionalen Feldes `ff` angefügt, usw.

a) Welche Feldlänge hat `f` ?

b) Das zweidimensionale Feld `ff` wird zerstört.

Durch welchen Zugriff kann man jetzt im eindimensionalen Feld `f` auf ein Element des Feldes `ff` zugreifen, das vor der Zerstörung an der Stelle `ff[i][j]` abgespeichert wurde ?

Bemerkungen:

In den obigen Aufgaben ist unter *i*-ter Zeile bzw. *i*-ter Spalte natürlich **programmtechnisch** die *i*-1- te Zeile bzw. *i*-1- te Spalte gemeint.

Das heißt zum Beispiel, dass mit 1. Zeile bzw. 1. Spalte programmtechnisch die 0. Zeile bzw. 0. Spalte gemeint ist

Lösungen:

1) a)

```
w[3] = '\\0';  
printf("%s", w);
```

b)

```
for(i=0; i<3; i++)  
    printf("%c", w[i]);
```

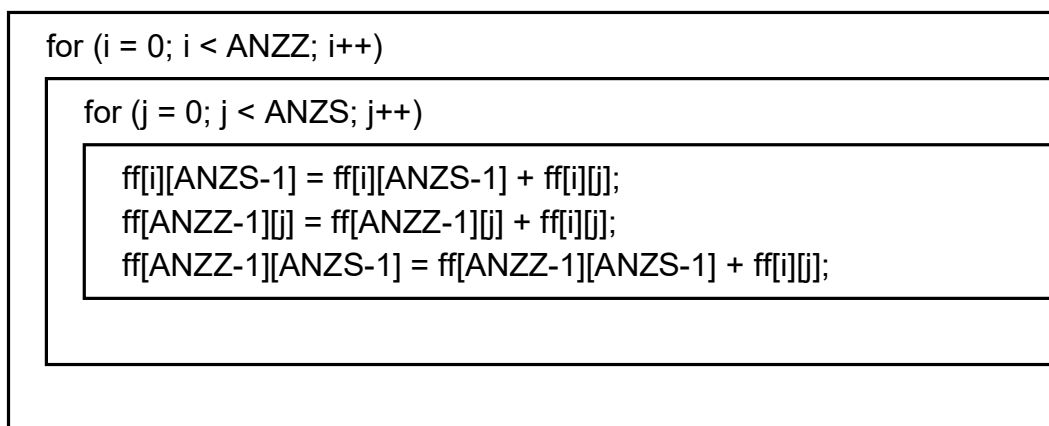
2)

Speicherplatzverbrauch von v1 = 10 * Speicherbedarf(int);

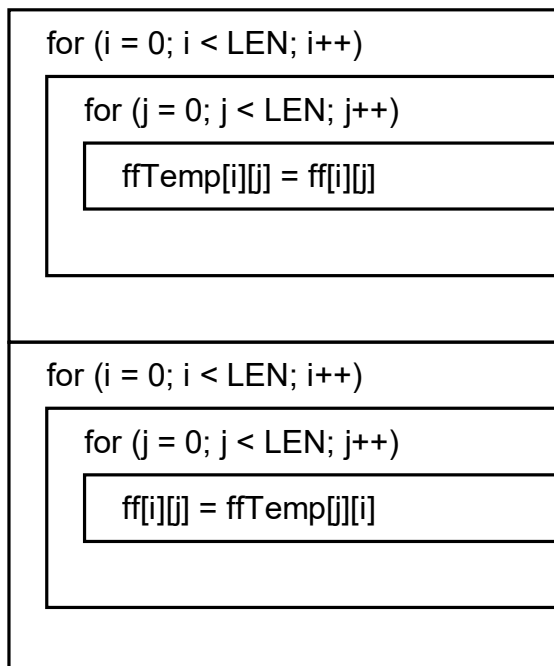
Speicherplatzverbrauch von v2 = 100 * Speicherbedarf(int);

Speicherplatzverbrauch von v3 = 1000 * Speicherbedarf(int);

3)



4)



5)

a)

Feldlänge f = ANZZ * ANZS

b)

ff[i * ANZS + j] = ff[i][j]

KLAUSUR 3 Programmierpraktikum 2BK11 21.2.2005 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 Punktabzug !!!!
- Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt
- 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:

- 1) Die Feldlänge soll vom Programmierer ohne aufwändige Bemühungen an genau einer Stelle leicht zu verändern sein (und muß dann neu kompiliert werden).
- 2) Programminfos verwenden (anwenderfreundliche Infos auf dem Bildschirm ausgeben!!).

AUFGABEN

1) Schreiben Sie ein C-Programm, das folgendes macht:

a) Alle Elemente eines Feldes (Datentyp: double, Länge zunächst einmal 5) sollen mit 0 vorbelegt werden (durch eine Schleife).

b) In das Feld sollen die Schulnoten einer Klasse eingegeben und im letzten Element des Feldes soll der berechnete Mittelwert aller Schulnoten abgespeichert werden.

c) Die eingegebenen Schulnoten und der Mittelwert sollen auf dem Bildschirm ausgegeben werden.

d) In ein zweidimensionales Feld (Datentyp double, Anzahl der Zeilen zunächst einmal 3, Anzahl der Spalten zunächst einmal 4) sollen die Endnoten in Mathematik, Englisch und Deutsch (erstes bzw. zweites Schulhalbjahr) eingegeben werden.

Im letzten Element einer jeden Zeile soll der berechnete Mittelwert aller Endnoten eines Schulhalbjahres berechnet und abgespeichert werden.

Im letzten Element einer jeden Spalte soll der berechnete Mittelwert eines jeden Fachs im ersten und zweiten Schulhalbjahres berechnet und abgespeichert werden.

Außerdem soll an geeigneter Stelle des Feldes noch der Mittelwert aller Fächer im ersten und zweiten Halbjahr abgespeichert werden.

e) Die eingegebenen Schulnoten und alle Mittelwerte sollen auf dem Bildschirm ausgegeben werden.

KLAUSUR 3 Programmierpraktikum 2BK11 28.2.2005 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
- 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:

- 1) Die Feldlänge soll vom Programmierer ohne aufwändige Bemühungen an genau einer Stelle leicht zu verändern sein (und muß dann neu kompiliert werden).
- 2) Programminfos verwenden (anwenderfreundliche Infos auf dem Bildschirm ausgeben!!).

AUFGABEN

1) Schreiben Sie ein C-Programm, das folgendes macht:

Gegeben ist ein zweidimensionales Feld ff (Datentyp: char) mit der Zeilenanzahl ANZR und der Spaltenanzahl ANZC. Zunächst gilt: ANZR = 3, ANZC = 5.

- a) Alle Elemente dieses zweidimensionalen Feldes sollen mit '\0' vorbelegt werden (durch eine Schleife).
- b) Geben Sie über Tastatur einen Text ein. Dieser Text muss in der 1. Zeile des Feldes abgespeichert werden. Falls der Text zu lang wird, wird er - ohne den Benutzer zu informieren - abgeschnitten.
- c) Kopieren Sie die 1. Zeile des Feldes in die 2. Zeile.
- d) Kopieren Sie den Text der 1. Zeile in umgekehrter Reihenfolge in die 3. Zeile.
- e) Geben Sie den Inhalt aller Zeilen auf dem Bildschirm aus (nach jeder Zeile muss ein Zeilenumbruch gemacht werden).

Bemerkungen:

In den obigen Aufgaben ist unter i-ter Zeile bzw. i-ter Spalte natürlich **programmtechnisch** die i-1-te Zeile bzw. i-1-te Spalte gemeint.

Das heißt zum Beispiel, dass mit 1. Zeile bzw. 1. Spalte programmtechnisch die 0. Zeile bzw. 0. Spalte gemeint ist

KLAUSUR 4 Programmiertheorie 2BKI1 9.6.2005 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 4 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 Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) Ein Anfangskapital wird zu einem bestimmten Zinssatz eine bestimmte Anzahl Jahre auf einem Konto angelegt, ohne die Zinsen abzuheben. Welchen Wert hat das Endkapital ?

a) Sie wollen dazu eine Funktion 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 !!!)

b) Da der "Programmierknecht" gerade ein größeres Projekt zu bearbeiten hat und sich deswegen irgendwo in der Südsee befindet, sind Sie gezwungen nach der von Ihnen entworfenen Leistungsbeschreibung die Funktion selbst zu programmieren. Schreiben Sie eine Funktion, das das Endkapital berechnet.

c) Rufen Sie diese Funktion in main auf.

Mathematische Anleitung anhand eines Beispiels:

Voraussetzungen: $K_0 = 100$ DM; $p = 1$ %;

n : Anzahl der Jahre,

z_n : neu hinzukommende Zinsen

K_n : Kapital nach n Jahren

n	z_n	K_n
0	0	100
1	$100 \cdot 0,01 = 1$	$100 + 1 = 101$
2	$101 \cdot 0,01 = 1,01$	$101 + 1,01 = 102,01$
3	$102,01 \cdot 0,01 = 1,0201$	$102,01 + 1,0201 = 103,0301$

Beispiel einer Dokumentation einer Funktion

```

/*****
/**
/**  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 mod∈{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:
  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
*/

```