

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) Was ist ein Algorithmus und welche Eigenschaften hat er ? (4 P)
- 2) Was ist die ANSI Norm ? (2 P)
- 3) Was ist ein Literal ? (2 P)
- 4) Was bedeutet Priorität in der Programmiersprache C ? (2 P)
Geben Sie ein Beispiel.
- 5) Was bedeutet Assoziativität in der Programmiersprache C ? (2 P)
Geben Sie ein Beispiel.
- 6) Kann die Ganzzahl 29999 in einer zwei Bytes grossen Integer-Zahl abgespeichert werden ?
Begründen Sie! (4 P)
- 7) Das Volumen einer Kugel berechnet sich wie folgt: (10 P)
$$V = \frac{4}{3} \cdot \pi \cdot r^3$$

Schreiben Sie das dazu passende C-Programm. Das Volumen muss in einer Variablen mit dem Datentyp float gespeichert werden. Die Konstante pi muss in einer Variablen abgespeichert werden.
- 8) (8 P)
 - a) Erscheint beim Kompilieren des folgenden syntaktisch korrekten Programmausschnitts eine Warnung ? Begründen Sie genau!
 - b) Welchen Wert hat f ?

```
...  
float f;  
f = 3/5*(3.5 + 4 * 2.5);  
...
```

9) Was gibt der folgende Programmausschnitt auf dem Bildschirm aus ? Begründen Sie! (16P)

Hilfestellung:

Werten Sie dazu jeweils die Ausdrücke (Bedingungen) in den if-Anweisungen aus.

```
-----  
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");  
    printf("Ausgabe6\n");  
if(x||b)  
    printf("Ausgabe7\n");  
if((x=2)&&8)  
    printf("Ausgabe8\n");  
-----
```

Bemerkung:

= = ist linksassoziativ

Lösung:

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

5) Assoziativität gibt an, in welcher Reihenfolge (von links nach rechts oder von rechts nach links) Operatoren mit der gleichen Priorität abgearbeitet werden.

Beispiel: * wird von links nach rechts abgearbeitet.

6) 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.

7)

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

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

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

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

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

    return 0;
}
```

8)

a) Es wird zuerst $3/5$ berechnet (Assoziativität von links nach rechts). Der Wert ist 0. 4 ist integer, 2.5 ist double. Also wird 4 in double 4.0 umgewandelt. Das Ergebnis 10.0 ist damit double. 3.5 ist double und damit ist $3.5 * 10.0$ auch double. Dieser Wert wird mit integer 0 multipliziert, wobei deshalb vorher die integer 0 in double 0.0 umgewandelt wird und den Wert double 0.0 ergibt.

Da double in float abgespeichert wird, wird die double Zahl 0.0 in float umgewandelt. Dabei kann ein Datenverlust entstehen. Deshalb gibt der Compiler eine Warnung aus.

b) 0.0

9)

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.

f) Ausgabe6: Nach `if(x=(2&&8))` steht keine Klammer {
Deshalb besteht der if-Körper der dieser if-Anweisung nur aus einer Anweisung. Die dieser Anweisung folgende Anweisung wird damit auf jeden Fall ausgeführt.

g) Ausgabe7: Der Wert von x ist 2 und der Wert von b ist 0. Damit wird 2 als wahr und 0 als falsch interpretiert. Also hat der Ausdruck $x \ || \ b$ den Wert 1 und ist damit wahr.

h) Ausgabe8: Der Wert des Ausdrucks (Zuweisungsausdrucks) $x=2$ ist 2. Der Wert 2 wird als wahr interpretiert. 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.

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) (5P)

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

2)

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: (20P)

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!

WICHTIG: EVA-Prinzip beachten.

3)

(25P)

Es soll der Ersatzwiderstand einer Parallelschaltung von 3 Widerständen berechnet werden. Wurde ein Widerstandwert kleiner oder gleich Null eingegeben, muß das Programm sofort beendet werden und eine entsprechende Meldung auf den Bildschirm ausgegeben werden. (insbesondere darf dann nicht mehr ein weiterer Widerstand eingegeben und der Ersatzwiderstand berechnet werden).

Dies muß mit dem EVA-Prinzip realisiert werden.

Außerdem muß in der Ausgabe noch angegeben werden, welcher Widerstandswert (z.B. 1. Widerstandwert) falsch eingegeben wurde.

Erstellen Sie das dazugehörige **Struktogramm**.

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.

WICHTIG: EVA-Prinzip beachten.

Lösung:

1)

a) switch: nur für ganzzahlige oder char Datentypen verwendbar

b) nichts

2)

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

4)

Eingabe r1								
W		r1 <= 0				F		
status = -1	Eingabe r2			F				
	W		r2 <= 0			F		
	status = -2	Eingabe r3		F				
		W		r3 <= 0			F	
		status = -3		status=0				
status==0						F		
rges = 1(1/r1+1/r2+1/r3)								
W		status == -1				F		
Ausgabe (1.Widerstand <=0)	status == -2					F		
	W		F					
	Ausgabe (2.Widerstand <=0)		status == -3			F		
			W		F			
		Ausgabe (3.Widerstand <=0)		Ausgabe(rges)				

KLAUSUR 1 Programmierpraktikum 2BK11 23.11.2010 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 !!!!
- Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

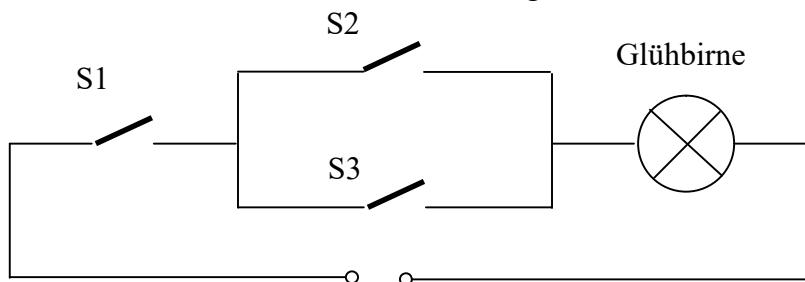
Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

1) Eine Glühbirne wird mit 3 Schaltern wie folgt verbunden:

25 P



Jeder der 3 Schalter (S1, S2 und S3) ist entweder „aus“ oder „ein“.

Schreiben Sie ein C-Programm, das ausgibt, ob die Glühbirne leuchtet oder dunkel ist.

Die Zustände der Schalter soll über Tastatur eingegeben werden.

Welcher Datentyp für den Zustand (ein / aus) gewählt wird, ist Sache des Programmierers.

Er legt fest, was „ein“ bzw. „aus“ bedeutet und sagt dies dem Anwender durch eine entsprechende Mitteilung auf dem Bildschirm.

(EVA-Prinzip beachten).

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

25 P

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.

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

Lösung:

1)

```
#include "stdafx.h"

int _tmain(int argc, _TCHAR* argv[]){
    int schalter1;
    int schalter2;
    int schalter3;
    int zustand;

    // Eingabe
    printf("Schalterstellung (0 / 1) Schalter 1 eingeben\n");
    scanf("%d",&schalter1);
    printf("Schalterstellung (0 / 1) Schalter 2 eingeben\n");
    scanf("%d",&schalter2);
    printf("Schalterstellung (0 / 1) Schalter 3 eingeben\n");
    scanf("%d",&schalter3);

    // Verarbeitung:
    zustand = schalter1 && (schalter2 || schalter3);

    // Ausgabe
    if (zustand == 0){
        printf("Die Lampe ist aus\n");
    }
    else{
        printf("Die Lampe ist ein\n");
    }

    return 0;
}
```

2)

```
#include "stdafx.h"

int _tmain(int argc, _TCHAR* argv[]){
    int note;
    printf("Bitte eine ganze Note eingeben\n");
    scanf("%d", &note);

    switch(note){
        case 1:
        case 2:
        case 3:
        case 4:
            printf("Prüfung bestanden \n");
            break;

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

        default:
            printf("Dies ist keine Note \n");
            break;
    }
    return 0;
}
```

KLAUSUR 1 Programmierpraktikum 2BK11 24.11.2010 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 !!!!
- Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

1)

50 P

Es soll ein Taschenrechner programmiert werden.

Zuerst muss dazu ein Zeichen über Tastatur eingegeben werden:

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

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

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

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

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

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

Bemerkungen:

1) Dies muß mit dem **EVA-Prinzip** realisiert werden. Jeden Teil entsprechend kommentieren mit: // Eingabe bzw. // Verarbeitung bzw. //Ausgabe

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

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

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

4) Durch 0 darf nicht dividiert werden.

Lösungen:

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

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

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

    switch(zeichen){
        case 'A':
        case 'a':
        case 'S':
        case 's':
        case 'M':
        case 'm':
        case 'D':
        case 'd':
            zustand = 0;
            break;
        default:
            zustand = -1;
    }

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

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

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

        else if(zeichen=='S' || zeichen=='s'){
            ergebnis = zahl1 - zahl2;
        }

        else if(zeichen=='M' || zeichen=='m'){
            ergebnis = zahl1 * zahl2;
        }

        else { // Division
            if(zahl2!=0){
                ergebnis = zahl1 / zahl2;
            }
            else{
                zustand = -2;
            }
        }
    }

    // A U S G A B E T E I L
    if(zustand==0){
```

```
        printf("Ergebnis=%f", ergebnis);
    }
    else if(zustand==-1){
        printf("Programmende\n");
    }
    else{ //Division durch 0
        printf("Durch 0 darf nicht dividiert werden\n");
    }
}
return 0;
}
```

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 auf jeder Seite unten rechts 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) 10P

Was ist allgemein der Hauptunterschied (bzgl. der Anzahl der Durchgänge) zwischen einer for-Anweisung und einer Do-Anweisung ?

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

2) 10P

Gegeben ist der folgende syntaktisch korrekte Programmteil:

```
i=10;
while(i<20); {
    printf("Hallo Welt\n");
}
```

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

3) a) 2P

Was ist eine Endlosschleife?

b) 3P

Schreiben Sie einen Programmausschnitt (in der Programmiersprache C) mit einer Endlosschleife.

4) 10P

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

5)

15P

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.

Lösungen:

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

Anzahl Durchgänge der while-Anweisung: ≥ 0

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

for(i = 0; i < -1; i++){
    printf("%d ", i);
}
```

2)

Kein einziges Mal, weil

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

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

Es wird endlos die leere Anweisung (also das Semikolon) ausgeführt.

Die Bildschirmausgabe wird nie erreicht!

```
while(i < 20)
;
{
    printf("Hallo Welt\n");
}
```

3)

a) Eine Endlosschleife wird nie verlassen

b)

```
while(1 == 1) {
}
```

4)

...

```
do {
    printf("ganze Zahl zwischen 1 und 5 eingeben\n");
    scanf("%d", &i);
} while(!(i >= 0 && i <= 5));
```

5)

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;
}
```

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) 25P

Erstellen Sie ein Struktogramm eines Programms, das die Fakultät einer ganzzahligen Zahl (≥ 1) berechnet.

$$n! = n * (n-1) * (n-2) * \dots * 1$$

Beispiele: $1! = 1$ $2! = 2 * 1$ $3! = 3 * 2 * 1$ $4! = 4 * 3 * 2 * 1$

Sie geben über Tastatur eine ganzzahlige Zahl (≥ 1) ein. Von dieser Zahl soll die Fakultät berechnet werden und das Ergebnis auf dem Bildschirm ausgegeben werden.

Beispiel:

Sie geben z.B. die Zahl 4 ein. Auf dem Bildschirm wird dann ausgegeben:

$$4! = 24$$

2) 25P

Zu Beginn eines Jahres wird ein Kapital zum Zinssatz p angelegt. Welchen Wert hat es nach 1, 2, 3, ... n Jahren, wenn die Zinsen auf dem Sparbuch bleiben?

Überlegen Sie, was der Anwender eingeben soll.

Erstellen Sie dazu das passende Struktogramm.

Beispiel:

Zinssatz: 10%

Anfangskapital: 1000 Euro

Zeitabschnitte	Kontostand
0	1000
1	$1000 + 10/100 * 1000 = 1000 + 100 = 1100$
2	$1100 + 10/100 * 1100 = 1100 + 110 = 1210$
3	$1210 + 10/100 * 1210 = 1210 + 121 = 1331$
...	...

KLAUSUR 3 Programmierpraktikum 2BK11 3.5.2011 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 !!!!
- Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

1) Die ersten anz2 Zahlen des 2. Feldes sollen an die ersten anz1 Zahlen des 1. Feldes angehängt werden. Dies soll mit der Funktion `anfuegen(...)` realisiert werden.

a) 20P

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

b) 20P

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

c) 10P

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

Testen Sie diese Funktion, indem die Elemente des Feldes (die zwei zusammengesetzten Felder) auf dem Bildschirm ausgegeben werden.

Lösung:

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

void anfüegen(const int feld1[], int anz1, const int feld2[], int anz2,
              int feld3[]);

int main() {
    int feld1[10]={1,2,3,4,5};
    int feld2[20]={6,7,8,9,10,11};
    int feld3[30];
    int i;
    anfüegen(feld1, 5, feld2, 6, feld3);
    for(i=0;i<11;i++){
        printf(" %d",feld3[i]);
    }
    return(0);
}
```

```
/*
**
** void anfüegen(const int feld1[], int anz1,
**               const int feld2[], int anz2, int feld3[]) **
**
**#
*/
Parameter:
    (i) const int feld1[] : das 1. Feld
    (i) int anz1          : die Anzahl der zu betrachtenden Elemente
                          des 1. Feldes
    (i) const int feld2[] : das 2. Feld
    (i) int anz2          : die Anzahl der zu betrachtenden Elemente
                          des 2. Feldes
    (o) int feld3[]       : das 3. Feld = anz1 Elemente von feld1 +
                          anz2 Elemente von feld2
```

Return:
kein

Beschreibung:
Fügt die ersten anz2 Elemente von feld2 an
die ersten anz1 Elemente von feld1 an.
Der Anwender dieser Funktion, muss sicherstellen, dass die
Felder genügend gross sind.
*/

```
void anfüegen(const int feld1[], int anz1, const int feld2[], int anz2,
              int feld3[]){
    int i;
    int j;

    for(i=0;i<anz1;i++){
        feld3[i]=feld1[i];
    }
    j=0;
    for(i=anz1;i<anz1+anz2;i++){
        feld3[i]=feld2[j];
        j++;
    }
}
```

Ersatzaufgabe

1)

20 P

Von einer Zahlenfolge soll die umgekehrte Zeichenfolge (in umgekehrter Richtung) bestimmt werden.

Beispiel:

-3, 5, -2, 9, 8 -----> 8, 9, -2, 5, -3

a) Implementieren Sie die Funktion "invertieren(...)" und dokumentieren diese (Leistungsbeschreibung) nach dem im Unterricht verwendeten Schema.

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

Testen Sie diese Funktion, indem die umgekehrte Reihenfolge auf dem Bildschirm ausgegeben wird.

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 auf jeder Seite unten rechts 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) 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) 20P
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) 20P
Da sich der Programmierknecht zur Zeit in einem sogenannten "Tschill-Urlaub" befindet und deshalb aktuell (und wohl auch zukünftig) nicht ansprechbar ist, muss die Funktion "kodieren" von Ihnen selbst implementiert werden. Implementieren Sie die Funktion "kodieren".
- c) 5P
Schreiben Sie ein Programm mit einem Aufruf der Funktion "kodieren" (keine Eingabe über scanf(), sondern Aufruf mit konkreten Parametern).
- d) 5P
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 Programmierpraktikum 2BK11 29.6.2011 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 !!!!
- Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe_Rechnernummer_Nachname_Vorname_Aufgabennr.
Beispiel: A_12_Mustermann_Erika_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

1)

a)

40P

Eines morgens in aller Frühe finden Sie auf Ihrem Schreibtisch die folgende Leistungsbeschreibung (Dokumentation) der Funktion "ersetzen (...)" vor (siehe Rückseite). Implementieren Sie diese Funktion.

Wo nötig (bzw. von Vorteil) den Bezeichner const verwenden.

b)

5P

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

Testen Sie diese Funktion, indem die Elemente der neuen Folge auf dem Bildschirm ausgegeben werden.

c)

5P

Welchen Nachteil hätte es, wenn man in der Funktion ersetze(...) dynamisch Speicherplatz für "neueFolge" belegt?

Man könnte ja die Größe dieses Speicherplatzes innerhalb der Funktion ersetze(...) berechnen lassen und dann genau so viel Speicherplatz für "neueFolge" allokalieren, wie "neueFolge" benötigt.

```

/*****
/**
/** void ersetzen(char zeichen, char folge[],
/** char ersetzung[], char neueFolge[])
/**
/**#*****
/*

```

Parameter:

- (i) char zeichen : zu ersetzendes Zeichen.
- (i) char folge[] : dort wird das "zeichen" ersetzt.
- (i) char ersetzung[]: Das Zeichen "zeichen" wird durch die Zeichenfolge "ersetzung" ersetzt.
- (o) char neueFolge[] : Die durch die Ersetzung entstehende neue Zeichenfolge

Return:

kein

Beschreibung:

In der Zeichenfolge "folge" wird beim erstmaligen Auftauchen des Zeichens "zeichen" dieses Zeichen durch die Zeichenfolge "ersetzung" ersetzt. Dadurch entsteht die neue Zeichenfolge "neueFolge".

Der Programmierer, der diese Funktion benutzt muß dafür Sorge tragen, dass beim Aufruf dieser Funktion genügend Speicherplatz für "neueFolge" bereitgestellt wird.

Beispiel 1:

zeichen: a
 folge: "raav"
 ersetzung: "xy"
 Nach dem Aufruf:
 neueFolge: "rxyav"

Beispiel 2:

zeichen: s
 folge: "raav"
 ersetzung: "xy"
 Nach dem Aufruf:
 neueFolge: "raav"

*/

Lösungen:

```
#include "stdafx.h"
#include <string.h>
#include <malloc.h>

void ersetzen(const char zeichen, const char folge[],
              const char ersetzung[], char neueFolge[]);

int main(int argc, char* argv[]){
    char folge[4]="raf";
    char ersetzung[3]="xy";
    char neueFolge[4];
    char zeichen='a';

    ersetzen(zeichen, folge, ersetzung, neueFolge);
    printf("neueFolge=%s\n",neueFolge);
    return 0;
}

void ersetzen(const char zeichen, const char folge[],
              const char ersetzung[], char neueFolge[]){
    int i=0;
    int j=0;
    int index=0;

    // Kopiere alle Elemente von folge bis zum Auftreten
    // des gefundenen Zeichens in neueFolge
    while(folge[i]!='\0' && folge[i]!=zeichen){
        neueFolge[i]=folge[i];
        i++;
    }
    // Zeichen wurde nicht gefunden
    if(i==strlen(folge)){
    }
    else{
        index=i;
        // Füge ersetzung an neueFolge an
        while(ersetzung[j]!='\0'){
            neueFolge[i]=ersetzung[j];
            i++;
            j++;
        }
        j = index+1;
        // Füge von j=index an die folge an die neue Folge an.
        while(folge[j]!='\0'){
            neueFolge[i]=folge[j];
            i++;
            j++;
        }
    }
    neueFolge[i]='\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 auf jeder Seite unten rechts 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) 5P

Eine Funktion $f(\dots)$ berechnet Wetterdaten und liefert diese in einem Feld zurück.

Da der aufrufende Programmierer, der diese Funktion benutzt, nicht weiß wie groß das Feld ist, das für die Wetterdaten verwendet wird, muß er zur Sicherheit immer Speicherplatz für ein großes Feld verwenden. Deshalb schreibt der Entwickler der Funktion f eine neue Funktion $f_{\text{Neu}}(\dots)$. In dieser wird nun dynamisch Speicher allokiert (mit malloc).

Da nach dem Aufruf dieser Funktion dieser Speicher bis zum Ende des Hauptprogramms zur Verfügung steht, gibt er ihn kurz vor Ende der Funktion (also eine Befehlszeile vor return) mit free wieder frei. Was ist an dieser Überlegung falsch?

2) 5P

In einer Funktion wird

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

Welcher Speicher steht jeweils

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

3) 5P

Etwas schwierig

In der Funktion $g(\dots)$ wird dynamisch Speicher für eine Zeichenfolge reserviert.

Wie muß der Parameter deklariert werden, damit nach dem Aufruf der Funktion g auf diesen zugegriffen werden kann?

4) Zu Beginn eines Zeitabschnitts wird ein Kapital zum Zinssatz p angelegt.
Welchen Wert hat es nach n Zeitabschnitten, wenn die Zinsen auf dem Sparbuch bleiben ?

Beispiel:

Zinssatz: 10%

Anfangskapital: 1000 Euro

Zeitabschnitte	Kontostand
0	1000
1	$1000 + 10/100 * 1000 = 1000 + 100 = 1100$
2	$1100 + 10/100 * 1100 = 1100 + 110 = 1210$
3	$1210 + 10/100 * 1210 = 1210 + 121 = 1331$
...	...

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

a) 20P
Entwerfen Sie für den "Programmierknecht" eine **Beschreibung** (Leistungsbeschreibung) der Funktion **vermehrten (...)** nach dem im Unterricht verwendeten Schema. (Kein Programm !!!)
Überlegen Sie sich zuerst genau, welche **Parameter** nötig sind und welche **Zusicherungen** Sie machen (möglichst viel erlauben).
Wo nötig (bzw. von Vorteil) den Bezeichner `const` verwenden.

b) 20P
Da der sogenannte "Programmierknecht" urplötzlich über Nacht Ihre Vorgesetzte wurde, muss die Funktion **vermehrten (...)** von Ihnen selbst implementiert werden. Implementieren Sie die Funktion **vermehrten (...)**.

Lösungen:

1) Nach dem Beenden von `return` in der Funktion, wurde der Speicherplatz schon freigegeben.

Er steht also nach dem Beenden der Funktion nicht mehr zur Verfügung.

2)

a) während des Aufrufs der Funktion: lokale Variable, dynamischer Speicher, Datei

b) nach dem Aufruf im Hauptprogramm: dynamischer Speicher, Datei

c) nach dem Beenden des Hauptprogramms: Datei

3)

Variable	Adresse	Inhalt
q	0800	0100

Was passiert?

`g(&100) --> g(0800) --> *0800 = 0123 (vom BS) -->`

q	0800	0123
---	------	------

0123 H

0124 a

...

*/

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

```
void g(char **p);
```

```
int main(){
    char *q;
    g(&q);
    printf("%s\n",q);
    return 0;
}
```

4)

```
/*
**
**  double vermehren(const double anfangskapital,
**                  const double zinssatz, const int n)
**
**#
*/
```

Parameter:

- (i) const double anfangskapital >= 0 : Anfangskapital
- (i) const double zins >= 0: Zins
- (i) const int n >= 0: Anzahl der Zeitabschnitte

Return:

berechnetes Endkapital nach n Jahren

Beschreibung:

Zu Beginn eines Zeitabschnitts wird das Anfangskapital "anfangskapital" zum Zinssatz "zinssatz" n Zeitabschnitte angelegt, Daraus ergibt sich dann das Endkapital, wenn die Zinsen auf dem Sparbuch bleiben.

5)

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

```
double vermehren(const double anfangskapital, const double
                zins, const int n);
```

```
int main(int argc, char* argv[]){
    double anfangskapital = 100;
    double endkapital;
    double zins = 10;
    int n = 0;

    endkapital=vermehren(anfangskapital, zins, n);
    printf("Endkapital=%f\n",endkapital);
    return 0;
}
```

```
double vermehren(const double anfangskapital, const double
                zins, const int n){
    int i=0;
    double kapital;
    kapital = anfangskapital;

    for(i=1; i<=n; i++){
        kapital=kapital+kapital*zins/100;
    }
    return kapital;
}
```