

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

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

AUFGABEN

1) 3+3+3 P

- a) Welche 3 Möglichkeiten (3 Worte nennen) gibt es, einen Algorithmus darzustellen?
- b) Erklären Sie die Begriffe Syntax und Semantik?
- c) Was ist ein Compiler ?

2) 3P

- a) Welchen Wert hat die Variable z (Datentyp: integer) nach Ausführung der folgenden C#-Anweisung ? Begründen Sie !

`z = 1/4 * 4;`

b)

Wie viele Bit braucht man, um genau 32 verschiedene Zustände darzustellen ?

3) 5P

- a) Welchen Wert haben x und y nach Ausführung des folgenden Programms:
Geben Sie den Wert von x und y nach jeder Anweisung an.

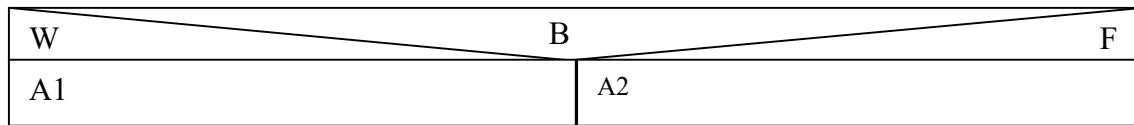
```
...  
x = 7;  
y = 3;  
x = x - y;  
y = x + y;  
x = y - x;  
...
```

- b) Was macht dieses Programm also im Endeffekt mit dem Inhalt der Variablen x und y (verbale Beschreibung) ?

4)

5P

Stellen Sie das folgende Struktogramm nur mit Hilfe von einseitigen Verzweigung als Struktogramm dar (das die gleiche Semantik hat).



5)

10P

Erstellen Sie ein Struktogramm, das das Minimum dreier Zahlen (Variablen z1, z2, z3) berechnet, in der Variablen min speichert und auf dem Bildschirm ausgibt.

EVA-Prinzip beachten!

6)

10P

Der folgende syntaktisch korrekte Teil eines C#-Programm soll die größte Zahl dreier (in z1, z2, z3 gespeicherter) Zahlen berechnen und in der Variablen max speichern.

Wenn das Programm korrekt ist, schreiben Sie "Der Algorithmus ist korrekt" und geben 5 Beispiele (mit konkreten Werten für z1, z2, z3 und dem berechneten Wert von max) an, wo er korrekt wird.

Wenn das Programm nicht korrekt ist, schreiben Sie "Der Algorithmus ist nicht korrekt" und geben ein **konkretes** Beispiel (mit konkreten Werten für z1, z2, z3 und dem berechneten Wert von max) an, bei dem der Algorithmus falsch wird.

```
... main(...) {
    ...
    // Eingabe (z1, z2, z3)
    max = z2;
    if (z1 < z2) {
        max = z1;
    }
    if (max < z3) {
        max = z3;
    }
    // Ausgabe (max)
    ...
}
```

7)

10P

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"

Bemerkung: Die Punktezahl kann auch nicht ganzzahlig sein!

Lösungen:

1)

a)

3P

Flussdiagramm, Struktogramm, Programm

b)

3P

Die Syntax definiert die äußeren Formgesetze dieser Programmiersprache (ähnlich den grammatikalischen Regeln einer natürlichen - wie z.B. der englischen- Sprache).

Die Semantik ist der Bedeutungsinhalt (ähnlich der Bedeutung der einzelnen Worte einer natürlichen - wie z.B. der italienischen - Sprache) der einzelnen Objekte einer Programmiersprache.

c)

3P

Ein Compiler ist ein Übersetzer, der einen in einer höheren Programmiersprache formulierten Text (ein sogenanntes Programm) in einen aus Maschinenbefehlen bestehenden Text (einem sogenannten Maschinenprogramm) verwandelt. Dieses kann dann vom Mikroprozessor abgearbeitet (ausgeführt) werden. Außerdem prüft er die Syntax.

2)

3P

a) $z = 0$, da 1 und 4 integer-Zahlen sind, deren Division den Wert 0 ergibt.

b) 5, da $32 = 2^5$

3)

5P

a) 3P

$x = 7;$

$y = 3;$

$x = x - y; \quad // \quad x = 7 - 3 = 4$

$y = x + y; \quad // \quad y = 4 + 3 = 7$

$x = y - x; \quad // \quad x = 7 - 4 = 3$

b) 2P

Das Programm vertauscht den Wert von x und y

4)

5P

W	B
A1	
W	nicht B
A2	

5)

10P

Eingabe(z1, z2, z3)	
W	$z1 < z2$
min = z1	min = z2
W	$z3 < \text{min}$
min = z3	
Ausgabe(min)	

6)

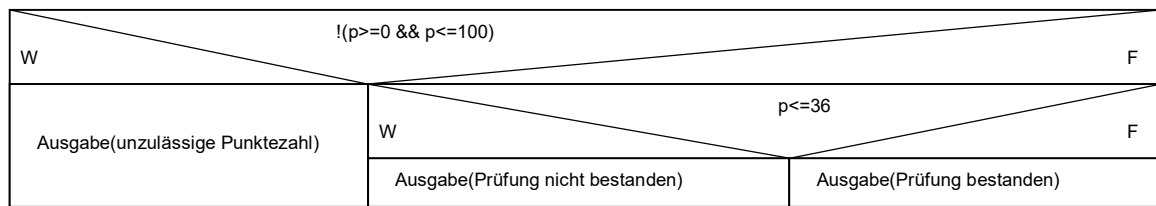
10P

Der Algorithmus ist nicht korrekt:

$z_1 = 30 \quad z_2 = 20 \quad z_3 = 10 \implies \max = 20$

7)

10P



KLAUSUR 1 Programmiertheorie 2BKI1 Nachtermin 1 Zeit: 90 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

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

AUFGABEN

1) 5P

- a) Wie viele Zustände genau kann man mit 8 Byte angeben? (als Hochzahl schreiben)
b) Näherungsweise Berechnung!

2) 10P

Simulieren Sie die folgende if-else Verzweigung durch eine oder mehrere einseitige Verzweigungen, wobei außer dem nicht Operator ! keine weiteren logischen Operatoren verwendet werden dürfen.

```
if (B1 && B2) {  
    A1  
}  
else{  
    A2  
}
```

3) 15P

Erstellen Sie ein Struktogramm, das drei Zahlen (Variablen z1, z2, z3) der Größe nach in aufsteigender Reihenfolge sortiert, in die Variablen klein, mittel, gross speichert und auf dem Bildschirm ausgibt. EVA-Prinzip beachten!

4)

20 P

Eine natürliche Zahl n ($n \geq 2$) heißt Primzahl, wenn sie nur durch 1 und sich selbst teilbar ist.
Beispiele für Primzahlen: 2, 3, 5, 7, 11, 13,

a) Erstellen Sie ein Flussdiagramm, das folgendes macht:

Es muß von einer eingegebenen ganzen Zahl $z \geq 2$ festgestellt werden, ob diese eine Primzahl ist. Das Ergebnis muß dann auf dem Bildschirm ausgegeben werden.

Tipp: Der Operator % berechnet den Rest bei einer Division. Damit kann man dann feststellen, ob eine Zahl eine andere teilt.

Beispiele:

$38 \% 3 = 2$, weil $38 : 3 = 12$ Rest 2, also teilt 3 nicht die Zahl 38

$38 \% 11 = 5$, weil $38 : 11 = 3$ Rest 5, also teilt 11 nicht die Zahl 38

$38 \% 2 = 0$, weil $38 : 2 = 19$ Rest 0, also teilt 2 die Zahl 38

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

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

AUFGABEN

Hilfsmittel: Eine selbstgeschriebene Dokumentation zu Methoden.

1) 5P

In einem C#-Programm wurde folgende Variable deklariert:

```
char [] v = new char [4];
```

Geben Sie Ihren Kommentar (falsch bzw. korrekt mit Begründung) zu folgenden

Anweisungen ab:

- a) `v[0] = 'v';` b) `v[4] = 'r';` c) `v[-1] = 'r';`
d) `v[1] = 3;` e) `v['f']='b';`

2) 10P

Gegeben sind die 2 folgenden Felder, in dem bestimmte Zeichen gespeichert sind.

```
char [] feld1 = new char [100000];
```

```
char [] feld2 = new char [100000];
```

Ziel ist es, den Inhalt von feld1 in umgekehrter Reihenfolge anzuordnen.

Beispiel:

In einem Feld v (der Länge 4) sind folgende Zeichen gespeichert:

In umgekehrter Reihenfolge sind sie so angeordnet:

'M'	'E'	'S'	'K'
'K'	'S'	'E'	'M'

Schreiben Sie ein Programm in C#, das Folgendes macht:

- a) Kopieren Sie den Inhalt von feld1 in umgekehrter Reihenfolge in feld2
b) Kopieren Sie den Inhalt von feld2 in feld1

Bem: Nur den Teil in `static void Main(string[] args)` machen, also kein using, namespace usw. verwenden.

Programm muß für eine beliebige Feldlänge (nicht nur für 100 000 funktionieren).

3)

10P

Ein Programm soll die Anzahl der Zeichen 'x' in dem Array "feld" berechnen und auf dem Bildschirm ausgeben.

Erstellen Sie dazu ein zugehöriges Struktogramm.

4)

25P

Die Methode berechneQuadrate(...) berechnet aus zwei Zahlen deren Quadratzahlen, also z.B. aus 3 und 5 wird $3^2 = 9$ und $5^2 = 25$ berechnet.

a) 10 P

Machen Sie zuerst eine Leistungsbeschreibung (Dokumentation) der Funktion berechneQuadrate(...) nach dem im Unterricht verwendeten Schema.

b) 10P

Implementieren Sie diese Funktion berechneQuadrate(...).

c) 5P

In main muß ein Aufruf der Funktion " berechneQuadrate(...)" realisiert werden, bei der die Quadratzahlen von 3 und 5 berechnet werden.

(keine Eingabe über Console.ReadLine(), sondern Aufruf mit diesen konkreten Parametern)

Lösungen:

3P

1)

```
v[0] = 'v';    // richtig
v[4] = 'r';    // falsch: nicht reservierter Speicher
v[-1] = 'r';   // falsch: das erste Feld beginnt bei 0
v[1] = 3;      // falsch: Feld hat de Datentyp char
v['f']='b';    // falsch: Index muß eine ganze Zahl >= 0 sein.
```

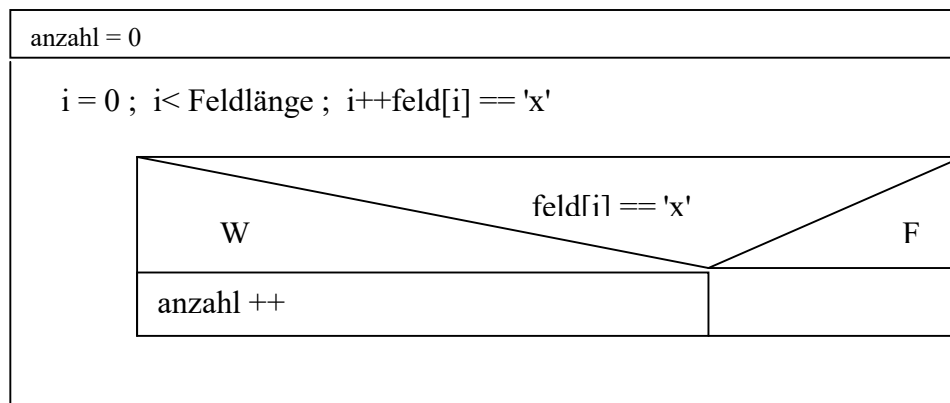
2)

```
static void Main(string[] args)
{
    int i;
    char[] feld1 = new char[100000];
    char[] feld2 = new char[100000];

    // feld1 in umgekehrter Reihenfolge nach feld2 kopieren
    for (i = 0; i < feld1.Length; i++)
    {
        feld2[i] = feld1[feld1.Length - i - 1];
    }

    // feld2 nach feld1 kopieren
    for (i = 0; i < feld1.Length; i++)
    {
        feld1[i] = feld2[i];
    }
}
```

3)



4)

```
/**
 *
 * double berechneQuadrate(double zahl1, double zahl2,
 *                           ref double zahl2hoch2)
 *
 */
/*#*****
```

Parameter:

- (i) double zahl1: die erste zu quadrierende Zahl
- (i) double zahl2: die zweite zu quadrierende Zahl
- (o) ref double zahl2hoch2: zahl2 * zahl2

Return:

- (o) zahl1 * zahl1

Beschreibung:

Berechnet aus den Zahlen "zahl1" und "zahl2" jeweils das Quadrat und gibt es über return bzw. zahl2hoch2 wieder zurück.

*/

```
static double berechneQuadrate(double zahl1, double zahl2,
                                ref double zahl2hoch2)
```

```
{
    double quadrat1;
    quadrat1 = zahl1 * zahl1;
    zahl2hoch2 = zahl2 * zahl2;
    return quadrat1;
}
```

```
static void Main(string[] args)
```

```
{
    double quadrat1 = 0;
    double quadrat2 = 0;
    double erg4 = 0;
    quadrat1 = berechneQuadrate(3, 5, ref quadrat2);
    Console.WriteLine("quadrat1= " + quadrat1 + " quadrat2="
                      + quadrat2);
}
```

KLAUSUR 2 Programmiertheorie 2BKI1 Nachtermin 1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

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

AUFGABEN

Hilfsmittel: Eine selbstgeschriebene Dokumentation zu Methoden.

1) 5P

In einem C#-Programm wird innerhalb dieses Programms die Länge eines Feldes verändert, weil das Feld zu klein ist und noch Platz für weitere Feldelemente benötigt wird.

Was meinen Sie dazu ?

Ist das möglich?

Begründen Sie bzw. nehmen Sie dazu Stellung.

2) 20P

Gegeben ist das folgende Feld, in dem Ihre Noten im Fach Mathematik gespeichert sind.

```
double [] noten = new double [100000];
```

Schreiben Sie ein Programm in C#, das Folgendes macht:

a) Berechnen Sie die Anzahl der Einsen, die Sie geschrieben haben.

b) Berechnen Sie den Mittelwert (Durchschnitt) der geschriebenen Noten.

Bem: Nur den Teil in static void Main(string[] args) machen, also kein using, namespace usw. verwenden.

Programm muß für eine beliebige Feldlänge (nicht nur für 100 000 funktionieren).

3)

25P

Die Methode `berechneMinMax(...)` berechnet aus zwei Zahlen das Minimum und das Maximum dieser 2 Zahlen.

a) 10 P

Machen Sie zuerst eine Leistungsbeschreibung (Dokumentation) der Funktion `berechneMinMax (...)` nach dem im Unterricht verwendeten Schema.

b) 10P

Implementieren Sie diese Funktion `berechneMinMax(...)`.

c) 5P

In `main` muß ein Aufruf der Funktion "`berechneMinMax (...)`" realisiert werden, bei der das Maximum von 17 und 13 berechnet wird.

(keine Eingabe über `Console.ReadLine()`, sondern Aufruf mit diesen konkreten Parametern)

Lösungen:

1)

5P

Die Länge eines Feldes muß genau ein einziges Mal festgelegt werden.

2)

20P

```
static void Main(string[] args)
{
    int i;
    int anzahlEinsen=0;
    double summe=0;
    double mittelwert;
    double [] noten = new double [100000];

    // a)
    for (i = 0; i<noten.Length; i++)
    {
        if(noten[i]==1)
        {
            anzahl++;
        }
    }
    // b)
    for (i = 0; i<noten.Length; i++)
    {
        summe=summe+noten[i];
    }
    mittelwert = summe / noten.Length;
}
```

3)

25P

```

/*****
/**
/**  double berechneMinMax(double zahl1, double zahl2,
/**                               ref double zahl2hoch2)
/**
/**
/*****
/

```

Parameter:

- (i) double zahl1: die erste Zahl
- (i) double zahl2: die zweite Zahl
- (o) ref double max: Minimum von zahl1 und zahl2

Return:

- (o) Maximum von zahl1 und zahl2

Beschreibung:

Berechnet aus den Zahlen zahl1 und zahl2 das Minimum von zahl1 und zahl2 und das Maximum von zahl1 und zahl2.

*/

```

static double berechneMinMax(double zahl1, double zahl2, ref min)
{
    double ergMax;
    if(zahl1<zahl2){
        min=zahl1;
        ergmax=zahl2;
    }
    else{
        min=zahl2;
        ergmax=zahl1;
    }
    return ergmax;
}

```

static void Main(string[] args)

```

{
    double mini = 0;
    double erg = 0;
    erg = berechneMinMax(17, 13, ref mini);
    Console.WriteLine("Min= " + mini + " Max=" + erg);
}

```

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

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

AUFGABEN

Hilfsmittel: Eine selbstgeschriebene Dokumentation zu Methoden.

1) 5P

In einem C#-Programm wurde folgende Variable deklariert:

```
char [] v = new char [4];
```

Geben Sie Ihren Kommentar (falsch bzw. korrekt mit Begründung) zu folgenden

Anweisungen ab:

- a) `v[0] = 'v';` b) `v[4] = 'r';` c) `v[-1] = 'r';`
d) `v[1] = 3;` e) `v['f']='b';`

2) 10P

Gegeben sind die 2 folgenden Felder, in dem bestimmte Zeichen gespeichert sind.

```
char [] feld1 = new char [100000];
```

```
char [] feld2 = new char [100000];
```

Ziel ist es, den Inhalt von feld1 in umgekehrter Reihenfolge anzuordnen.

Beispiel:

In einem Feld v (der Länge 4) sind folgende Zeichen gespeichert:

In umgekehrter Reihenfolge sind sie so angeordnet:

'M'	'E'	'S'	'K'
'K'	'S'	'E'	'M'

Schreiben Sie ein Programm in C#, das Folgendes macht:

- a) Kopieren Sie den Inhalt von feld1 in umgekehrter Reihenfolge in feld2
b) Kopieren Sie den Inhalt von feld2 in feld1

Bem: Nur den Teil in `static void Main(string[] args)` machen, also kein using, namespace usw. verwenden.

Programm muß für eine beliebige Feldlänge (nicht nur für 100 000 funktionieren).

3)

10P

Ein Programm soll die Anzahl der Zeichen 'x' in dem Array "feld" berechnen und auf dem Bildschirm ausgeben.

Erstellen Sie dazu ein zugehöriges Struktogramm.

4)

25P

Die Methode berechneQuadrate(...) berechnet aus zwei Zahlen deren Quadratzahlen, also z.B. aus 3 und 5 wird $3^2 = 9$ und $5^2 = 25$ berechnet.

a) 10 P

Machen Sie zuerst eine Leistungsbeschreibung (Dokumentation) der Funktion berechneQuadrate(...) nach dem im Unterricht verwendeten Schema.

b) 10P

Implementieren Sie diese Funktion berechneQuadrate(...).

c) 5P

In main muß ein Aufruf der Funktion " berechneQuadrate(...)" realisiert werden, bei der die Quadratzahlen von 3 und 5 berechnet werden.

(keine Eingabe über Console.ReadLine(), sondern Aufruf mit diesen konkreten Parametern)

Lösungen:

3P

1)

```
v[0] = 'v';    // richtig
v[4] = 'r';    // falsch: nicht reservierter Speicher
v[-1] = 'r';   // falsch: das erste Feld beginnt bei 0
v[1] = 3;      // falsch: Feld hat de Datentyp char
v['f']='b';    // falsch: Index muß eine ganze Zahl >= 0 sein.
```

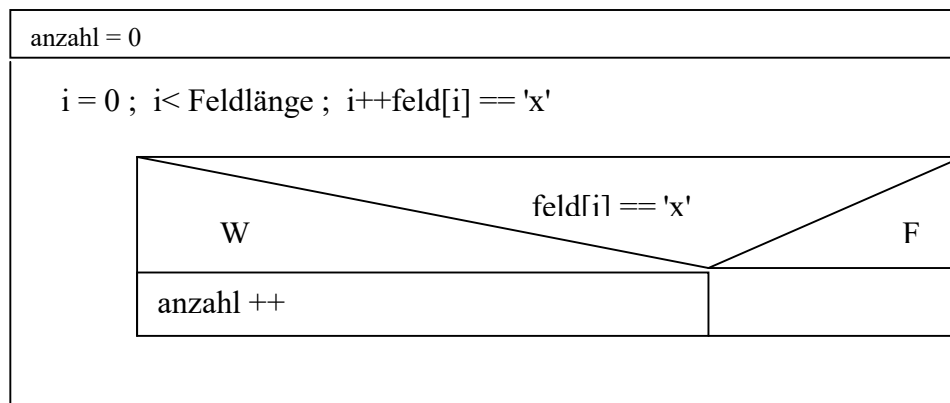
2)

```
static void Main(string[] args)
{
    int i;
    char[] feld1 = new char[100000];
    char[] feld2 = new char[100000];

    // feld1 in umgekehrter Reihenfolge nach feld2 kopieren
    for (i = 0; i < feld1.Length; i++)
    {
        feld2[i] = feld1[feld1.Length - i - 1];
    }

    // feld2 nach feld1 kopieren
    for (i = 0; i < feld1.Length; i++)
    {
        feld1[i] = feld2[i];
    }
}
```

3)



4)

```

/*****
/**
/**  double berechneQuadrate(double zahl1, double zahl2,
/**                               ref double zahl2hoch2)
/**
/**
/*****
/*#*****
/*

```

Parameter:

- (i) double zahl1: die erste zu quadrierende Zahl
- (i) double zahl2: die zweite zu quadrierende Zahl
- (o) ref double zahl2hoch2: zahl2 * zahl2

Return:

- (o) zahl1 * zahl1

Beschreibung:

Berechnet aus den Zahlen "zahl1" und "zahl2" jeweils das Quadrat und gibt es über return bzw. zahl2hoch2 wieder zurück.

*/

```

static double berechneQuadrate(double zahl1, double zahl2,
                               ref double zahl2hoch2)

```

```

{
    double quadrat1;
    quadrat1 = zahl1 * zahl1;
    zahl2hoch2 = zahl2 * zahl2;
    return quadrat1;
}

```

```

static void Main(string[] args)

```

```

{
    double quadrat1 = 0;
    double quadrat2 = 0;
    double erg4 = 0;
    quadrat1 = berechneQuadrate(3, 5, ref quadrat2);
    Console.WriteLine("quadrat1= " + quadrat1 + " quadrat2="
                      + quadrat2);
}

```

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

AUFGABEN

1) 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: // jeweils 3P

(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: // 5P

Kodiert "anzahl" Zahlen der Zahlenfolge "zahlen", indem der Wert "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 3 Programmiertheorie 2BKI1 Nachtermin 2 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

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

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

AUFGABEN

1)

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" allokieren, 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';
}
```