Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

**INE1**

# Musteraufgaben für die Semesterendprüfung

| | |
|---|---|
| Hilfsmittel | Vier Seiten selbst verfasste Zusammenfassung<br>• keine weiteren Hilfsmittel<br>• keine elektronischen Hilfsmittel |
| Abgabe | Füllen Sie das erste Aufgabenblatt aus und schreiben Sie **alle Lösungsblätter mit Ihrem Namen, Vornamen und Ihrer Klasse** sowie der entsprechenden **Aufgabennummer** an. **Geben Sie alle Aufgaben- und Lösungsblätter ab.** |
| Hinweise | Lesen Sie alle Aufgabenstellungen sorgfältig durch, bevor Sie mit der Bearbeitung der ersten Aufgabe beginnen.<br><br>Schreiben Sie Ihre Lösungen direkt auf die Aufgabenblätter. |
| Maximum | 64 Punkte |
| Zeit | 90 Minuten |

## Aufgabe 1: Fallunterscheidungen 7 Pt

Schreiben Sie eine Funktion (nur diese!), die aufgrund von einem Datum (Tag,Monat) feststellt, in welcher Jahreszeit dieses Datum liegt.
Der Funktion werden der Tag und der Monat als Parameter (short int) übergeben. Die Funktion gibt dann die Jahreszeit in folgender Form zurück: 1 für Frühling, 2 für Sommer, 3 für Herbst und 4 für Winter.

Sie können voraussetzen, dass das Datum gültig und korrekt ist, das müssen also Sie nicht extra prüfen. Beim Monat entspricht 1 dem Januar, 2 dem Februar, usw.,. Die Jahreszeiten beginnen am 21. März, 21. Juni, 23. September bzw. 21. Dezember. Die Tage, an denen eine Jahreszeit beginnt, sollen der beginnenden Jahreszeit zugerechnet werden: also der 21. März dem Frühling usw.

Definieren und benützen Sie *symbolische Konstanten* für die Jahreszeiten!

## Aufgabe 2: 1D-Array / Einfache Berechnungen 7 Pt

Gesucht ist eine Funktion *berechneAnteile(..)*, deren Deklaration wie folgt aussieht:

*void berechneAnteile(int werte[], double prozent[], int n);*

Der Funktion werden ein int-Array mit Werten und ein double-Array für die prozentualen Anteile übergeben. Beide Arrays sind gleich gross, die Anzahl der Elemente wird als dritter Parameter n übergeben. Die Funktion soll nun die prozentualen Anteile der Elemente des Arrays *werte* berechnen und in das Array *prozent* eintragen.

**Beispiel 1:**

```
int zahlen[2] = {3, 3};
double prozent[2];
berechneAnteile(zahlen, prozent, 2);
// prozent ist dann { 50.0, 50.0 }
```

**Beispiel 2:**

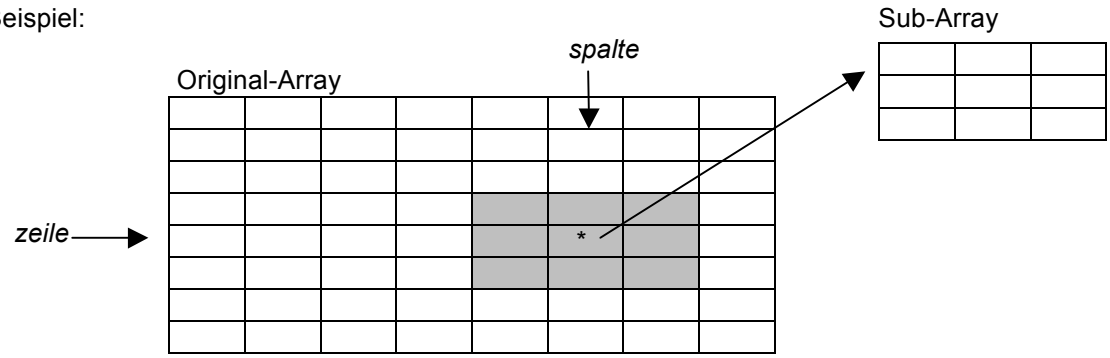
```
int zahlen[4] = {5, 5, 15, 0};
double prozent[4];
berechneAnteile(zahlen, prozent, 4);
// prozent ist dann { 20.0, 20.0, 60.0, 0.0 }
```

Implementieren Sie die Funktion *berechneAnteile(..)*:

## Aufgabe 3: 2D-Arrays 9 Pt

Aus einem 2D-Array (Original-Array genannt) soll ein Sub-Array "herausgeschnitten" werden.
Es wird ein Array-Element (*zeile,spalte*) gewählt. Der Sub-Array umfasst dieses Element und seine Nachbar-Elemente.

Beispiel:




Der Original-Array ist gegeben und sieht wie folgt aus:

```
#define ROWS 8
#define COLS 8

int origArray[ROWS][COLS] = { {1,6,5,3,8,9,4,7},
                              {10,11,16,15,13,18,19,20},
                              {21,26,25,23,28,29,30,37}
                               usw.

                             };
```

Schreiben Sie ein Programm-Fragment, welches aus dem Original-Array die entsprechenden Werte - gemäss obiger Abbildung - in einen neuen Array (Sub-Array genannt) kopiert.
Die Definition der noch notwendigen Variablen gehört auch dazu.
Es muss geprüft werden, ob der Sub-Array innerhalb des Original-Arrays liegt. Wenn das nicht der Fall ist, wird eine Fehlermeldung ausgegeben und es wird nichts kopiert. Andernfalls wird kopiert.

## Aufgabe 4: File I/O 9 Pt

Ein Programm generierte eine *Textdatei*
"`noten.txt`" mit folgendem Inhalt:

```
Meier       4.8
Frei        5.5
Haeberling  3.7
Hug         5.4
Hegi        3.2
Steiner     5.6
Leuenberger 4.1
Rechsteiner 5.9
```

*Ergänzen Sie direkt auf dem Aufgabenblatt* den nachfolgenden Programmcode so, dass dieses Programm die Textdatei *zeilenweise einliest*, zur Kontrolle *beim Lesen jede Zeile wieder ausgibt* und anschliessend den *Notendurschnitt der Klasse berechnet und ausgibt*.

Detaillierte Spezifikationen:

a) Der Name der Datei ist im Programm fix kodiert ("`C:\noten.txt`"). Die Datei kann *beliebig viele Zeilen* haben.
b) Wenn die Datei nicht gefunden wird, dann soll die Programmausführung mit Fehlermeldung abgebrochen werden.
c) Beim Lesen einer Zeile werden der Studentennamen und die Note temporär in den Variablen `stud` und `note` speichert (`fscanf` verwenden). Diese Variablen werden zur Kontrolle sofort wieder ausgegeben.
   Das Programm merkt sich also nur den Inhalt einer Zeile und verarbeitet sofort diese Informationen. Es findet keine Speicherung aller Studenten und ihren Noten in irgendwelchen Arrays etc.
d) Beim Lesen der Zeilen müssen die Noten zusammenaddiert werden (Variable `summe`) und nach der letzten Zeile soll dann der Notendurchschnitt berechnet und ausgegeben werden.

Beispiel der Programm-Ausgabe (für die obige Datei `noten.txt`; Ihre Version darf auch weniger schöne Formatierung erzeugen):

```
Meier                   4.80
Frei                    5.50
Haeberling              3.70
Hug                     5.40
Hegi                    3.20
Steiner                 5.60
Leuenberger             4.10
Rechsteiner             5.90

 Notendurchschnitt =    4.78
```

**Programmcode zum Ergänzen:**

```
#include <stdio.h>
#include <stdlib.h>

int main(void){
  char stud[50];    // Speicher für den momentan eigelesenen Studentenname
  float note;       // Speicher für die momentan eigelesene Note
  float summe;      // Für die Notendurchschnittsberechnung
  int zaehler;      // zählt Anzahl Studenten in der Datei


  // Hier kommt Ihr Code:

  // Datei zum Lesen vorbereiten und Fehlermeldung
  // ausgeben, wenn Datei nicht vorhanden
```

```
  // ganze Datei zeilenweise lesen, ausgeben und die Summe der Noten
  // berechnen
```

```
  // Durchschnitt der Klasse berechnen und ausgeben
```

```
}
```

## Aufgabe 5: Structs / Array of Structs 11 Pt

Ein Programm soll eine Notenliste für einen Kurs verwalten. Jeder Eintrag der Liste besteht aus folgenden Komponenten:

- `kurzzeichen`, C-String mit maximal 8 Zeichen, zum Beispiel "mustepet"
- `note`, Notendurchschnitt ungerundet, Datentyp *double*
- `noteGerundet`, Notendurchschnitt gerundet auf halbe Noten, Datentyp *double*

Eine erste einfache Testversion des Programms soll nur die Daten von der Tastatur einlesen, den Notendurschnitt auf halbe Noten runden, am entsprechenden Ort eintragen, und dann alle Daten wieder ausgeben. Ein Teil des Programms ist im Folgenden angegeben:

```
#define MAXTEILNEHMER 50

struct ... /* Teilaufgabe a */

int main(void)
{
    ... /* Teilaufgabe b */
    int anzahl;

    anzahl = eingabeNotenListe(notenListe, MAXTEILNEHMER);
    ausgabeNotenListe(notenListe, anzahl);

    return 0;
}

int eingabeNotenListe(struct Eintrag notenListe[], int max)
{
    ... /* Teilaufgabe c */
}

void ausgabeNotenListe(struct Eintrag notenListe[], int anzahl)
{
    ...
}
```

Ein Beispieldialog mit dem Programm könnte wie folgt aussehen. Die fettgedruckten Teile werden über die Tastatur eingegeben, der Rest wird vom Programm ausgegeben. Es werden zwei Datensätze eingegeben und der Eingabe-Vorgang dann mit einem Punkt abgeschlossen, was die Funktion *eingabeNotenListe* beendet:

```
Naechster Eintrag (Abschluss durch Punkt):
Kurzzeichen:        mustepet
Notendurchschnitt:  4.6

Naechster Eintrag (Abschluss durch Punkt):
Kurzzeichen:        bkrt
Notendurchschnitt:  4.3

Naechster Eintrag (Abschluss durch Punkt):
Kurzzeichen:        .
```

**Aufgaben:**

a. Wie lautet die Deklaration der Datenstruktur für einen Eintrag? (2P)

b. Am Anfang der Funktion *main* soll nun eine Variable *notenListe* als Array mit maximal MAXTEILNEHMER Einträgen definiert werden. Geben Sie die Variablendefinition an. (2P)

c. Ein Teil der Funktion *eingabeNotenListe* ist im Folgenden angegeben. Die Variable *anzahl* dient dazu, die tatsächliche Anzahl der eingegebenen Datensätze zu bestimmen. Ergänzen Sie die fehlenden Teile der Funktion *eingabeNotenListe.* (5P)

```
int eingabeNotenListe(struct Eintrag notenListe[], int max)
{
    int i, anzahl=0;
    for (i=0; i<max; i++)
    {
        printf("Naechster Eintrag (Abschluss durch Punkt):\n");
        printf("Kurzzeichen:        ");
        // Eingabe
        if (notenListe[i].kurzzeichen[0] == '.')
            break;
        printf("Notendurchschnitt:  ");
        // Eingabe
        notenListe[i].noteGerundet
            = floor(notenListe[i].note * 2 + 0.5)/2.0;
        printf("\n");
    }
    return anzahl;
}
```

d. In eingabeNotenListe wird der gerundete Wert für die Note automatisch eingetragen. Dieser Wert wird in einem etwas komplizierten Ausdruck bestimmt. Welcher Wert wird eingetragen für die folgenden Werte von notenListe[i].note? (2P)

| notenListe[i].note | notenListe[i].noteGerundet |
|---|---|
| 4.8 | |
| 5.7 | |

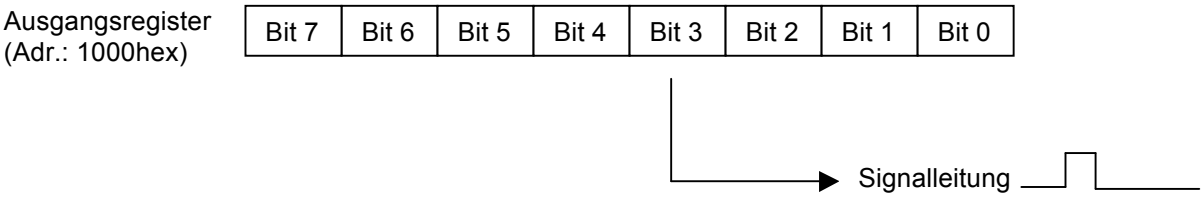
Hinweis: Aus der Beschreibung der Funktion floor in math.h:

```
double floor ( double x );
```
Round down value
Returns the largest integral value that is not greater than x.

## Aufgabe 6: Bit-Manipulation 6 Pt

Auf einer digitalen Signalleitung soll ein Impuls ausgegeben werden:
(Die Länge des Impulses ist nicht von Bedeutung)

Dazu wird ein Bit in einem Ausgangs-Register (an der Adresse 1000 Hexadezimal) entsprechend manipuliert:




a) Simulieren Sie zuerst diesen Vorgang, indem Sie anstatt eines Registers eine Variable "c" vom Typ *unsigned char* entsprechend manipulieren.

Schreiben Sie ein Code-Fragment, das auf der simulierten Signalleitung (Bit Nr. 3) diesen Impuls erzeugt.
*Alle übrigen Bit-Werte* in dieser Variablen *dürfen* bei dieser Aktion nicht *verändert werden*.

**Lösung:**

b) Schreiben Sie eine zweite Version, in welcher ein Hardware-Register an der Adresse 1000hex in gleicher Weise manipuliert wird. Tip: Pointers.

**Lösung:**

## Aufgabe 7: Endlosschleife 4 Pt

Die Schleife im folgenden Codefragment ist eine *Endlosschleife*.

```
double f(double x);
void zeichnePunkt(double x, double y);
...
double x=-1;
do {
      zeichnePunkt(x,f(x));
      x=x+0.1;
} while (x!=0);
...
```

a) *warum*?
Begründen Sie das mit ein bis zwei kurzen Sätzen oder Stichworten:

b) welche *Abhilfe* für dieses Problem schlagen Sie vor?

## Aufgabe 8: Darstellung ganzer Zahlen 3 Pt

Geben Sie die Darstellung der Zahl $9753_{10}$ in der Basis 16 („hexadezimal“), 8 („oktal“) und 2 („binär“) an.

## Aufgabe 9: Pointers 8 Pt

Gegeben ist das folgende Programm:

```
#include <stdio.h>
#define MAX_WOERTER 10

int main()  {
   int * p1, * p2;
   int r1, r2, r3, r4;

   int x = 10;
   int y = 20;
   int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    char * woerter [MAX_WOERTER] = {"Hallo", "Alois", "Winterthur",
                                    "Aprilscherz", NULL};

   p1 = &y;
   p2 = a;

   r1 = *p1 - *p2;
   r2 = *p2+5 + y + *p1;
   r3 = *(p2 + 5) + x;

   p2++;
   r4 = p2[3];

   printf ("resultat1 = %d \n", r1);
   printf ("resultat2 = %d \n", r2);
   printf ("resultat3 = %d \n", r3);
   printf ("resultat4 = %d \n", r4);
}
```

**a) Welche Werte werden hier ausgegeben?** (4 Pt)

resultat1 = …

resultat2 = …

resultat3 = …

resultat4 = …

**b)** Welche der folgenden Anweisungen sind korrekt und schreiben als Ausgabe "Winterthur" auf dem Bildschirm? **Kreuzen Sie die richtige Antwort an.** (4 Pt)

| | | | |
|---|---|---|---|
| **i.** | `printf ("resultat5 = %s \n", woerter[2]);` | korrekt ☐ | falsch ☐ |
| **ii.** | `printf ("resultat5 = %s \n", * woerter[2]);` | korrekt ☐ | falsch ☐ |
| **iii.** | `printf ("resultat5 = %s \n", & woerter[2][0]);` | korrekt ☐ | falsch ☐ |
| **iv.** | `printf ("resultat5 = %s \n", woerter[2][0]);` | korrekt ☐ | falsch ☐ |