Übung 1

1 Aufgabe

Nennen Sie 5 Beispiele für Algorithmen aus der Alltagswelt.

2 Aufgabe

Nennen Sie 5 Beispiele von Hochsprachen. Sind diese Sprachen jeweils Compiler- oder Interpreter-basiert? Gibt es Mischformen? Wenn ja, wie sehen diese aus?

3 Aufgabe

Nennen Sie Vor- und Nachteile des Compiler- und Interpreter-basierten Ansatzes. Was bieten Mischformen?

4 Aufgabe

Bringen Sie das "Hallo Welt"-Programm aus der Vorlesung auf Ihrer bevorzugten Plattform zum Laufen. Skizzieren Sie dabei den Ablauf und vergleichen Sie mit den Schritten, die in der Vorlesung durchgeführt wurden.

Übung 2

1 Aufgabe

Schreiben Sie ein Programm, das Ihren Vor- und Nachnamen sowie Ihr Alter ausgibt.

2 Aufgabe

Was passiert mit dem "Hallo Welt"-Programm, wenn Sie ein Semikolon entfernen? Erklären Sie.

3 Aufgabe

Vielleicht wird Ihnen der Compiler bei der Übersetzung Ihres Programms Fehler melden. Was halten Sie dabei für besser:

- dass der Compiler nach dem ersten gefundenen Fehler mit einer wortreichen Erklärung abbricht und Ihnen so die Möglichkeit einer schnellen Reparatur gibt
- dass der Compiler versucht, möglichst viele Fehler zu entdecken, d.h. nicht nach jedem Fehler abbricht?

4 Aufgabe

4.1 Teil 1

Was kann der Compiler mit unbekannten Wörtern tun, d.h. Wörtern, die nicht in der Sprachbeschreibung erklärt sind?

4.2 Teil 2

Was passiert, wenn die Zeile

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

fehlt? Erklären Sie die Ausgabe des Compilers.

5 Aufgabe

Macht es in C einen Unterschied, wieviele Zeilen oder Leerzeichen Sie in Ihrem Programm verwenden?

6 Aufgabe

Warum gibt es eigentlich einen Präprozessor? Ist er bei allen Sprachen notwendig?

7 Aufgabe

Auch in der Mathematik gibt es Variablen. Wie unterscheiden sie sich von den Variablen in Programmiersprachen?
Wie z.B. müßte in der Mathematik die folgende Formel für eine Integer-Variable gedeutet werden und wie in der Informatik:

$$i = i + 1$$

8 Aufgabe

Was geschieht bei folgender Code-Strecke? Kommentieren Sie jede Zeile einzeln.

```
int main() {
...
    int i;
    i = 42;
...
}
```

9 Aufgabe

Wieso ist die folgende Zeile in C nicht gültig:

```
    int i;
    42 = i;
```

10 Aufgabe

Verändern Sie das “Hallo Welt” Programm so, dass es

1. “Hallo schöne Welt” ausgibt
2. “Hallo” und “Welt” in verschiedenen Zeilen ausgibt
3. zwischen “Hallo” und “Welt” ein Tab Abstand existiert
4. verschiedene fest vorgegebene Zeile ausgeben kann

Was müßten Sie tun, dass das Programm eine Zeile, die der Aufrufer eingibt, auch wieder zurückgibt?

11 Aufgabe

Was wird durch die Angabe eines **Datentyps** eigentlich gesagt? Warum muß man in vielen Programmiersprachen einen Typ angeben? Geht es auch ohne Datentypen?

12 Aufgabe

Was bedeutet eigentlich “Gleitpunktzahl einfacher oder doppelter Genauigkeit”? Wie wird solch eine Zahl im Rechner dargestellt? Wie wird eine Integer-Zahl dargestellt?

13 Aufgabe

Wie kann das ’A’ eigentlich auf einem Computer dargestellt werden, der doch nur 0 und 1 kennt? Was wird im Rechner abgelegt für die Integer-Zahl 1 und für das Zeichen ’1’.

Übung 3

1 Aufgabe

Ändern Sie das Ausgangsprogramm so ab, dass es auch **int** und **float** sowie einzelne **char** ausgeben kann. Was müssen Sie dafür tun?

2 Aufgabe

Bringen Sie das folgende Programm zum Laufen:

```
#include <stdio.h>

int main() {
  int a, b;
  double d;
  char ch;

  i = 4;
  a = 5.;
  b = 3;
  d = 4.5;
  ch = "A";

  printf("%d\n", a);
  printf("%f\n", b)
  printf("%5.2f\n", d);
  printf("%c\n, ch);


  return(0);
}
```

Welche Bedeutung hat die Format-Anweisung "5.2f"? Recherchieren Sie. Gibt es weitere Format-Anweisungen?

3 Aufgabe

Wie hängen die Begriffe **Anweisung** und **Ausdruck** mit den Begriffen **r-value** und **l-value** zusammen?

4 Aufgabe

Schreiben Sie ein Programm, das zwei Integer-Zahlen einliest und sowohl die Summe als auch das Produkt ausgibt. Das gleiche soll auch für float-Zahlen gebaut werden.

5 Aufgabe

Schreiben Sie ein Programm, dass für 10 verschiedene DM-Werte von 1 bis 5 DM die entsprechenden Euro-Werte errechnet und ausgibt. Woher können die DM-Werte kommen? Implementieren Sie zwei verschiedene Möglichkeiten. Welche Datentypen verwenden Sie?

6 Aufgabe

Schreiben Sie ein Programm, das für drei verschiedene Radien zwischen 2 und 3 jeweils die Oberfläche der Kugel und das Volumen berechnet und ausgibt. Die Radien können fest kodiert sein.

7 Aufgabe

Welche Darstellung haben die folgenden Ausdrücke in C:

$$(a + b)/(c - d) - de$$

und

$$\sqrt{3}(a + b)$$

8 Aufgabe

Was passiert jeweils bei den folgenden Zeilen. Welche Werte haben jeweils i, k, l. Erklären Sie.

```
int i, k, l;
i = 42;

k = i++;
l = ++i;
```

9 Aufgabe

Welchen Sinn machen die folgenden Ausdrücke:

```
int a, b, d;
a & 0707;
b | 0xC;
a = a ^ a;
b = b << 3;
d = d & ~07;
```

10 Aufgabe

Für welchen Wert der Integer-Variablen x ist der folgende Ausdruck wahr: `((x >= 10) && (x <= 200)) || (x == 5) && (x==6)`

11 Aufgabe

Betrachten Sie den folgenden Ausdruck: `(x < 0) || ((++x > 5) && (x < 7))`. Welchen Wert hat x nach Auswertung dieses Ausdrucks, wenn anfangs `x = -1;` war?

12 Aufgabe

Welche Werte haben die einzelnen Konstanten in der folgenden Aufzählung? Ist das immer so? Erklären Sie.

```
enum hugo {AA, BB = 3, CC='c', DD};
```

13 Aufgabe

Was passiert, wenn Sie eine Variable als `const` zu deklarieren und dann in Ihrem Programm verändern? Probieren Sie es aus!

14 Aufgabe

Welche Operatoren kennen Sie, die mit Zeichenketten arbeiten? Was machen Sie z.B., wenn Sie zwei Zeichenketten aneinander hängen wollen und was, wenn Sie Teile aus einer Zeichenkette verwenden wollen?

15 Aufgabe

Ändern Sie das Programm zur Umrechnung von DM in Euro so ab, dass es die umzuwandelnden Werte einliest und nur im Falle eines Wertes größer als 1000DM auch umwandelt. Sie wollen sich doch nicht mit Peanuts abgeben, oder?

16 Aufgabe

Berechnen Sie für einen eingegebenen Radius (Float) die Oberfläche und das Volumen einer Kugel mit diesem Radius. Beachten Sie, dass dies nur für positive Radien definiert ist.

17 Aufgabe

Wie sehen die Fluß-Diagramme für die ein-armige und zwei-armige **if**-Anweisung aus? Erklären Sie.

18 Aufgabe

Schreiben Sie ein Programm, das zwei Integer einliest und dafür das Maximum bestimmt. Verwenden Sie dafür

1. nur die ein-armige If-Anweisung
2. die zwei-armige `if-else` Anweisung

19 Aufgabe

Betrachten Sie folgendes Code-Segment:

```
if (a > 5)
   if (a < 10)
      b = c;
else
   printf("a kleiner 5");
```

Was soll diese Code-Strecke ausdrücken? Was passiert hier tatsächlich? Haben Sie Verbesserungsvorschläge?

20 Aufgabe

Sie wollen einen Wert a einsortieren in die 5er-Intervalle von 5 aufwärts. Sie codieren dazu:

```
if (a > 5)
    printf(">5\n");
  else if (a > 10)
    printf(">10\n");
  else if (a > 15)
    printf(">15\n");
  else
    printf("a kleiner 5\n");
```

Was ist dazu zu sagen? Haben Sie Verbesserungsvorschläge?

Übung 4

1 Aufgabe

Schreiben Sie ein Programm, das 5 Buchstaben einliest und je nach Buchstabe unterschiedliche Zeichenketten ausgibt. Das Programm soll 3 von Ihnen gwählte Zeichen explizit verarbeiten können und den Rest durch einen gemeinsamen String beantworten.
Verwenden Sie hierfür einmal die `if-else`-Form und dann die `switch`-Anweisung. Welche ist hier sinnvoller?

2 Aufgabe

Schreiben Sie ein Programm, das einen Integer-Wert einliest und bei gerader Zahl das Quadrat ausgibt, bei ungerader die Wurzel. Verwenden Sie die switch-Anweisung. Das Programm soll nur die Zahlen zwischen 1 und 14 akzeptieren. Was machen Sie bei einer fehlerhaften Eingabe?

3 Aufgabe

Welche Werte haben a und b nach dem Durchlauf der folgenden Schleifen. Die Initial-Werte seien $a = 5$ und $b = 0$.

```
while (a >= b) if (b==0) a--;
while (a >= b) {a=a-b; if (b==0) b++;}
```

4 Aufgabe

Schreiben Sie ein Programm, das Zeichen von der Eingabe liest. Das Programm soll auf die Eingabe eines “s” und eines “S” sich beenden. Andere Konsonanten sind als Fehler abzulehnen. Das letzte eingelesene Zeichen soll jeweils ausgegeben werden.

5 Aufgabe

Schreiben Sie ein Programm, das nur Zahlen von der Eingabe liest und die Summe ausgibt. Das Programm soll enden, wenn eine Null eingeben wird.

6 Aufgabe

Macht die folgende Code-Strecke einen Sinn und wenn ja, welchen?

```
do {
  /* do something important */
} while (0);
```

7 Aufgabe

Wie sieht das Fluß-Diagramm der **for**-Schleife aus?

8 Aufgabe

Was passiert bei folgender **for**-Schleife?

```
for (i = 0; i < 10; i++) i = i - 1;
```

9 Aufgabe

Schreiben Sie ein Programm, das die Summe der ersten hundert Integer-Werte ausgibt.

10 Aufgabe

Schreiben Sie ein Programm, das den größten gemeinsamen Teiler (ggT) zweier Integer-Zahlen u und v berechnet. Verwenden Sie dabei die Tatsache, daß, wenn u größer als v ist, der ggT durch den ggT von v und $u - v$ gegeben ist.

11 Aufgabe

Schreiben Sie ein Programm, das die Euler'sche Zahl e bis zu einer vorgegebenen Genauigkeit berechnet. Verwenden Sie dabei die Tatsache, dass gilt

$$e = \lim_{n \to \infty} a_n$$

mit

$$a_n = \sum_{i=0}^{n} \frac{1}{n!}$$

Dabei ist $n! = n * (n-1)! = 1 * 2 * ... * n$ die Fakultät von n.

12 Aufgabe

Schreiben Sie ein Programm, das - mittels **getchar()** - Zeichen aus der Eingabe liest und dabei zählt wieviel Ziffern, Zwischenräume und andere Zeichen dabei vorkamen. Dieses Ergebnis soll ausgegeben werden.

13 Aufgabe

Was ist der Unterschied zwischen den beiden folgenden Code-Strecken:

```
for (i = 0; i < 10; i ++) {
  if ( i % 2 == 0)
     continue;
  printf("%d: Hier\n", i);
}
```

und

```
for (i = 0; i < 10; i ++) {
  if ( i % 2 == 0)
     break;
  printf("%d: Hier\n", i);
}
```

14 Aufgabe

Was geschieht bei folgender Code-Strecke:

```
for (i = 0; i < 10; i++) {
  switch (i) {
    case 1: case 2: case 3:
        continue;
    case 4:
        break;
    default:
        break;
  }
  printf("%d: neuer Durchlauf\n", i);
}
```

Übung 5

1 Aufgabe

In der Vorlesung haben Sie die Struktur der Deklaration von Funktionen kennengelernt. Dabei sahen Sie, dass die Argument-Liste aus Paaren (Typ,Name) besteht.
In manchen Fällen kann jedoch der Name des Arguments weggelassen werden. Wann? Wieso? Begründen Sie.

1.1 Lösung

Wenn es nur um die Struktur der Funktion geht, z.B. bei einer (Vorwärts-)Deklaration. Der Compiler braucht nur die **Signatur**.

2 Aufgabe

Wie sieht die Definition der kleinst-möglichen Funktion aus?

2.1 Lösung

bla(){}
denn **int** als Return-Wert, wenn nichts gesagt wird

3 Aufgabe

Was passiert bei folgendem Programm und warum? Was sollte geschehen?

```
#include <stdio.h>

void myfunction(int a) {
  printf("Der Wert von a = %d", a);
}

int main() {
  int a = 42;
  myfun(a);
  return(0);
}
```

3.1 Lösung

Compiler-Fehler, da myfun nicht deklariert.

4 Aufgabe

Schreiben Sie die Deklarationen für folgende Funktionen:

1. Rückgabe von **float**, Eingabe von **int**, **int**, **char**
2. Rückgabe keines Wertes, Eingabe von **float**, **int**
3. Rückgabe von **int**, keine Eingabe.

Muß die Anzahl der Argumente immer bekannt sein? Was meinen Sie?

4.1 Lösung

variable Anzahl von Argumenten nur mittels Bibliothek, *stdarg*.

5 Aufgabe

Was ist eigentlich das allgegenwärtige

```
int main() {
  ...
  return(0);
}
```

5.1 Lösung

eine normale Funktion, d.h. die main-Variablen sind auch nur lokale Variablen
Unterschied: die Ausführung beginnt bei main

6 Aufgabe

Was passiert in C bei folgender Code-Strecke:

```
void f (int a) {
  a++;
}
...
```

```
int c = 4;
f(c);
printf("Der Wert von c: %d\n", c);
```

Was bedeutet das für die Veränderung von Werte in einer Funktion? Womit arbeiten Sie im Falle des **call-by-value** also faktisch? Was geschähe im Falle eines **call-by-reference**?

6.1 Lösung

Arbeit immer mit **lokaler Kopie** des Wertes printf gibt 4 zurück. call-by-reference: 5

7 Aufgabe

Schreiben Sie ein modulares Programm, das bei Eingabe dreier Integer das Maximum dieser Zahlen ermittelt und ausgibt.

7.1 Lösung

```
#include <stdio.h>

int max3(int x, int y, int z) {
  int max;

  if (x < y)
    max = y;
  else
    max = x;

  if (z > max)
    max = z;

  return max;
}

int main() {
  int a, b, c;

  scanf("%d",&a);
```

```
   scanf("%d",&b);
   scanf("%d",&c);

   printf("Das Maximum = %d\n", max3(a,b,c));

   return(0);
}
```

8 Aufgabe

Schreiben Sie das ggT-Programm modular, d.h. verwenden Sie Funktionen.

9 Aufgabe

Teilen Sie das Programm zur Bestimmung des Maximums dreier Zahlen in mehrere Dateien auf. Welche sind hier sinnvoll? Wie erzeugen Sie daraus ein korrektes und funktionsfähiges Programm?

9.1 Lösung

Header:

```
int max3(int, int, int)
Implementierung:
Code
Main:
#include "max.h"
....
```

10 Aufgabe

Kann man in C Funktionen auch innerhalb anderer Funktionen definieren? Wie unterscheidet sich das von Variablen?

10.1 Lösung

Nein, in C sind Funktionen nur außerhalb anderer Funktionen zu definieren. Damit sind Funktionen immer **extern**. Dieses Schlüsselwort - und auch **static** innerhalb anderer Funktionen - sind für Funktionen damit nicht möglich/nötig.

11 Aufgabe

Was passiert bei folgender Code-Strecke? Welchen Wert hat *var* nach Aufruf der Funktion. Erklären Sie.

```
int var = 5;
...
void func() {
  int var;
  var++;
  return var;
}
```

11.1 Lösung

Das globale *var* wird in der Funktion durch das lokale *var* überdeckt. Wenn also nicht durch Zuweisung `var = func(var);` verändert wird, behält *var* den Wert 5.

12 Aufgabe

Schreiben Sie ein Programm, das eine Funktion beinhaltet, die ausgibt, wie oft sie aufgerufen wurde. Verwenden Sie dazu keine globale Variable.

12.1 Lösung

verwende **static int** als Counter innerhalb der Funktion

13 Aufgabe

Betrachten Sie folgende Aufruf-Hierarchie:

```
int f3(int a) {
  return a;
}

int f2 (int a, int b) {
  f3(a);
}

int f1(int a, int b, int c) {
```

```
    f2(a,b);
  }
```

Wie sieht der Stack graphisch für diese Aufruf-Folge aus, wenn gerade die Funktion **f3** aktiv ist.

13.1 Lösung

Reihenfolge ist von oben nach unten wie der Stack – logisch gesehen – selbst.

1. Frame für f3:
   (a) Rücksprungadresse von/in f2
   (b) weitest links-stehender ("erster") Parameter, d.h. a
2. Frame für f2:
   (a) Rücksprungadrsse von/in f1
   (b) weitest links-stehender ("erster") Parameter, d.h. a
   (c) zweiter Parameter, d.h. b
3. Frame für f1: analog

14 Aufgabe

Berechnen Sie die Fakultät einer eingegebenen Integer-Zahl durch eine Rekursion. Die Fakultät $f(n) = n!$ eines ganzzahligen Wertes ist definiert durch

$$f(n) = n * f(n-1) f(0) = 1;$$

14.1 Lösung

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

long long int fak(int n) {
  long long int w;

  if (n == 0) {
    return 1;
  }
```

```
  w = n * fak(n-1);

  return(w);
}

int main() {

  int a;

  printf("Bitte Eingabe des Wertes:\n");
  scanf("%d",&a);

  printf("Die Fakult"at von %d ist %lld\n", a, fak(a));

  return(0);
}
```

15 Aufgabe

Schreiben Sie das Programm zur Bestimmung des größten gemeinsamen Teilers auf eine rekursive Form um und modularisieren Sie es in Header- und Implementierungsdateien.

15.1 Lösung

```
 /* ggt.h */
#include <stdio.h>
#include <math.h>
int ggt(int a, int b);

 /* ggt.c */
#include "ggt.h"
int ggt(int a, int b) {
  int w;
  if (a == b)
    return a;

  if (a < b)
    w = ggt(b-a,a);
  else
```

```
    w = ggt(a-b,b);

  return w;
}

 /* driver.c */
#include "ggt.h"
int main() {
  int u, v;
  int i;
  //  int ggt;

  printf("Eingabe der ersten Zahl: \n");
  scanf("%d", &u);
  printf("Eingabe der zweiten Zahl: \n");
  scanf("%d", &v);

  if ((u <= 0) || (v <= 0)) {
    printf("Fehler\n");
    return(-1);
  }

  printf("Der ggT ist %d\n", ggt(u,v));

  return (0);
}
```

16 Aufgabe

In der Vorlesung wurde behauptet, dass Rekursionen mitunter langsamer sind als Iterationen. Können Sie dafür eine Erklärung finden?

16.1 Lösung

Hintergrund: Rekursionen haben mehr Overhead bei den vielen Funktionsaufrufen, s. Auf- und Abbau der Stack-Frames (u.a. Kopieren der Argumente etc.)

Übung 6

1 Aufgabe

Erst Datentypen geben den "rohen" Bitfolgen im Hauptspeicher des Rechners eine Bedeutung. Was ergibt z.B. die Bitfolge 100001, wenn man sie als Integer bzw. als Zeichen interpretiert?

2 Aufgabe

Welche Aktionen muß der Compiler für die folgenden Operationen durchführen?

```
int i;
long l;
float f;
double d;

char c;

i += l;
l += i;

l = i*f;
f = l*f;

d = f *d;

c = i;
i = c;
```

3 Aufgabe

Warum kann ein Einsatz von **typedef** sinnvoll sein?

4 Aufgabe

Deklarieren Sie die folgenden Felder:

1. ein Feld von 10 Integern
2. ein Feld von 20 Charactern
3. ein Feld von 6 Feldern aus je 4 Integer-Elementen

und greifen Sie auf jedes fünfte der vorhandenen Elemente zu.

5 Aufgabe

Erklären Sie mit eigenen Worten zeilenweise das Verhalten des Programms

```
#include <stdio.h>

int main() {
  char in[10] = "Hallo Welt";

  printf("%s\n", in);

  return(0);
}
```

6 Aufgabe

Wieso wird zwischen dem Zeichen 'c' und der Zeichenkette "c" unterschieden? Erklären Sie damit die Äquivalenz, die in der Vorlesung behauptet wurde:

```
char astr[] = "Hallo\n";   /* "aquivalent */
char bstr[] = {'H', 'a', 'l', 'l', 'o', '\n', '\0'};
```

7 Aufgabe

Das Skalarprodukt zweier Vektoren x und y des R^n kann in Komponenten-Schreibweise geschrieben werden als

$$x \cdot y = \sum_{i=0}^{n} x_i y_i$$

Schreiben Sie ein Programm, dass für zwei Vektoren des R^{11} das Skalarprodukt berechnet. Verwenden Sie dazu Felder und eine Berechnungsfunktion.

8 Aufgabe

Schreiben Sie ein Programm, dass aus einer Reihe von Zahlen (integer oder float) die Größte heraus sucht. Verwenden Sie dazu ein Feld gegebener Größe, das die Zahlen speichert und eine Funktion, in der die Bestimmung stattfindet.

9 Aufgabe

Schreiben Sie ein Programm, das die Länge einer Zeichenkette ausgibt. Die Länge soll dabei **nicht** der verantwortlichen Funktion angegeben werden

10 Aufgabe

Schreiben Sie ein Programm, das eine Zeichenkette bis zu einer maximalen Länge einliest und in umgekehrter Reihenfolge wieder ausgibt. Verwenden Sie dazu Funktionen und Felder.

11 Aufgabe

Schreiben Sie ein Programm, das eine einzugebene Folge von Integer-Zahlen der Reihe nach sortiert und wieder ausgibt. Sie können dazu den einfachen Bubble-Sort verwenden. Schauen Sie ggf. in der Literatur nach.

12 Aufgabe

Felder als Funktionsargumente benötigen etwas Aufmerksamkeit, da die Länge nicht durch den Namen des Arguments gekennzeichnet ist. Wie steht es denn bei der Rückgabe eines Feldes? Geht das überhaupt, was haben Sie dabei zu beachten?

13 Aufgabe

Deklarieren Sie die folgenden **structs**:

1. eine struct aus einer Integer-Variable, einem Integer-Feld der Länge 2 und einem Float

2. eine struct aus einer Double, einem Character-String der Länge 20 und der struct aus Punkt 1

3. ein Feld der Länge 10 aus structs von Punkt 1

4. eine struct aus einem float, einem int und einem Feld von 4 structs des Typs zwei

Können Strukturen überhaupt Strukturen als Komponenten besitzen? Wieso wird hierbei eigentlich immer nur von der Deklaration gesprochen?

14 Aufgabe

Ist der folgende Code korrekt? Begründen Sie.

```
int aint;
struct bla {
  int aint;
  float fl;
};
```

15 Aufgabe

Schreiben Sie ein Programm, das Bücher-Bestellungen entgegennehmen kann. Ein Buch soll dabei durch den Titel, den Autor, der ISBN und der Anzahl der Seiten beschrieben werden. Diese Angaben sollen solange der Eingabe des Anwenders entnommen werden, bis der Anwender ein Ende wünscht.

16 Aufgabe

Vergleichen Sie Felder und Strukturen bzgl. der folgenden Eigenschaften:

1. neuer Datentyp

2. explizite Initialisierung

3. Vergleichs-Operator (==)

4. Zuweisung, d.h. $a = b$

5. Funktionsargument und Rückgabewert

17 Aufgabe

Wie groß sind die jeweiligen Datentypen? Prüfen Sie Ihre Antwort in einem Programm nach.

```
struct dop{
  int a;
  int b[2];
  float f;
};

union au {
  int i;
  double d;
};
```

18 Aufgabe

Betrachten Sie die folgende Verwendung einer **union**.

```
union ttt {
  int i;
  float f;
};
union ttt uvar;
...
int a = uvar.i;
```

Was ist dabei problematisch? Wie kann man das Problem lösen?

19 Aufgabe

Skizzieren Sie den generellen Aufbau eines C-Programms. Wie werden die verschiedenen Bestandteile des Programms auf unterschiedliche Dateien verteilt? Was muß dabei alles beachtet werden?

Übung 7

1 Aufgabe

Erst Datentypen geben den "rohen" Bitfolgen im Hauptspeicher des Rechners eine Bedeutung. Was ergibt z.B. die Bitfolge 100001, wenn man sie als Integer bzw. als Zeichen interpretiert?

2 Aufgabe

Welche Aktionen muß der Compiler für die folgenden Operationen durchführen?

```
int i;
long l;
float f;
double d;

char c;

i += l;
l += i;

l = i*f;
f = l*f;

d = f *d;

c = i;
i = c;
```

3 Aufgabe

Deklarieren Sie die folgenden Felder:

1. ein Feld von 10 Integern
2. ein Feld von 20 Charactern
3. ein Feld von 6 Feldern aus je 4 Integer-Elementen

und greifen Sie auf jedes fünfte der vorhandenen Elemente zu.

4 Aufgabe

Erklären Sie mit eigenen Worten zeilenweise das Verhalten des Programms

```
#include <stdio.h>

int main() {
  char in[10] = "Hallo Welt";

   printf("%s\n", in);

  return(0);
}
```

5 Aufgabe

Das Skalarprodukt zweier Vektoren x und y des R^n kann in Komponenten-Schreibweise geschrieben werden als

$$x \cdot y = \sum_{i=0}^{n} x_i y_i$$

Schreiben Sie ein Programm, dass für zwei Vektoren des R^{11} das Skalarprodukt berechnet. Verwenden Sie dazu Felder und eine Berechnungsfunktion.

6 Aufgabe

Schreiben Sie ein Programm, dass aus einer Reihe von Zahlen (integer oder float) die Größte heraus sucht. Verwenden Sie dazu ein Feld gegebener Größe, das die Zahlen speichert und eine Funktion, in der die Bestimmung stattfindet.

7 Aufgabe

Schreiben Sie ein Programm, das die Länge einer Zeichenkette ausgibt. Die Länge soll dabei **nicht** der verantwortlichen Funktion angegeben werden

8 Aufgabe

Schreiben Sie ein Programm, das eine Zeichenkette bis zu einer maximalen Länge einliest und in umgekehrter Reihenfolge wieder ausgibt. Verwenden Sie dazu Funktionen und Felder.

9 Aufgabe

Deklarieren Sie die folgenden **structs**:

1. eine struct aus einer Integer-Variable, einem Integer-Feld der Länge 2 und einem Float
2. eine struct aus einer Double, einem Character-String der Länge 20 und der struct aus Punkt 1
3. ein Feld der Länge 10 aus structs von Punkt 1
4. eine struct aus einem float, einem int und einem Feld von 4 structs des Typs zwei

Können Strukturen überhaupt Strukturen als Komponenten besitzen? Wieso wird hierbei eigentlich immer nur von der Deklaration gesprochen?

10 Aufgabe

Ist der folgende Code korrekt? Begründen Sie.

```
int aint;
struct bla {
  int aint;
  float fl;
};
```

11 Aufgabe

Schreiben Sie ein Programm, das Bücher-Bestellungen entgegennehmen kann. Ein Buch soll dabei durch den Titel, den Autor, der ISBN und der Anzahl der Seiten beschrieben werden. Diese Angaben sollen solange der Eingabe des Anwenders entnommen werden, bis der Anwender ein Ende wünscht.

12 Aufgabe

Wie groß sind die jeweiligen Datentypen? Prüfen Sie Ihre Antwort in einem Programm nach.

```
struct dop{
  int a;
  int b[2];
  float f;
};

union au {
  int i;
  double d;
};
```

13 Aufgabe

Betrachten Sie die folgende Verwendung einer **union**.

```
union ttt {
  int i;
  float f;
};
union ttt uvar;
...
int a = uvar.i;
```

Was ist dabei problematisch? Wie kann man das Problem lösen?

14 Aufgabe

Skizzieren Sie den generellen Aufbau eines C-Programms. Wie werden die verschiedenen Bestandteile des Programms auf unterschiedliche Dateien verteilt? Was muß dabei alles beachtet werden?

1 Aufgabe

Nennen Sie 5 Beispiele für Algorithmen aus der Alltagswelt.

2 Aufgabe

Nennen Sie 5 Beispiele von Hochsprachen. Sind diese Sprachen jeweils Compiler- oder Interpreter-basiert? Gibt es Mischformen? Wenn ja, wie sehen diese aus?

3 Aufgabe

Nennen Sie Vor- und Nachteile des Compiler- und Interpreter-basierten Ansatzes. Was bieten Mischformen?

4 Aufgabe

Bringen Sie das "Hallo Welt"-Programm aus der Vorlesung auf Ihrer bevorzugten Plattform zum Laufen. Skizzieren Sie dabei den Ablauf und vergleichen Sie mit den Schritten, die in der Vorlesung durchgeführt wurden.

5 Aufgabe

5.1 Teil 1

Experimentieren Sie mit den Escape-Sequenzen und stellen Sie "Hallo Welt" auf verschiedene Weisen dar. Was passiert bei **\q** und warum? Was müssen Sie tun, um den "Backslash" \ darzustellen?

5.2 Teil 2

Lassen Sie das Programm den folgenden Text ausgeben:
`Das `` sind Anfuehrungszeichen.`

6 Aufgabe

Schreiben Sie ein Programm, das Ihren Vor- und Nachnamen sowie Ihr Alter ausgibt.

7 Aufgabe

Was passiert mit dem "Hallo Welt"-Programm, wenn Sie ein Semikolon entfernen? Erklären Sie.

8 Aufgabe

Vielleicht wird Ihnen der Compiler bei der Übersetzung Ihres Programms Fehler melden. Was halten Sie dabei für besser:

- dass der Compiler nach dem ersten gefundenen Fehler mit einer wortreichen Erklärung abbricht und Ihnen so die Möglichkeit einer schnellen Reparatur gibt
- dass der Compiler versucht, möglichst viele Fehler zu entdecken, d.h. nicht nach jedem Fehler abbricht?

9 Aufgabe

9.1 Teil 1

Was kann der Compiler mit unbekannten Wörtern tun, d.h. Wörtern, die nicht in der Sprachbeschreibung erklärt sind?

9.2 Teil 2

Was passiert, wenn die Zeile

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

fehlt? Erklären Sie die Ausgabe des Compilers.

10 Aufgabe

Macht es in C einen Unterschied, wieviele Zeilen oder Leerzeichen Sie in Ihrem Programm verwenden?

11 Aufgabe

Warum gibt es eigentlich einen Präprozessor? Ist er bei allen Sprachen notwendig?

12 Aufgabe

Auch in der Mathematik gibt es Variablen. Wie unterscheiden sie sich von den Variablen in Programmiersprachen?
Wie z.B. müßte in der Mathematik die folgende Formel für eine Integer-Variable gedeutet werden und wie in der Informatik:

$$i = i + 1$$

13 Aufgabe

Was geschieht bei folgender Code-Strecke? Kommentieren Sie jede Zeile einzeln.

```
int main() { ... int i; i = 42; ... }
```

14 Aufgabe

Wieso ist die folgende Zeile in C nicht gültig:

```
int i; 42 = i;
```

15 Aufgabe

Verändern Sie das "Hallo Welt" Programm so, dass es

1. "Hallo schöne Welt" ausgibt
2. "Hallo" und "Welt" in verschiedenen Zeilen ausgibt
3. zwischen "Hallo" und "Welt" ein Tab Abstand existiert
4. verschiedene fest vorgegebene Zeile ausgeben kann

Was müßten Sie tun, dass das Programm eine Zeile, die der Aufrufer eingibt, auch wieder zurückgibt?

16 Aufgabe

Was wird durch die Angabe eines **Datentyps** eigentlich gesagt? Warum muß man in vielen Programmiersprachen einen Typ angeben? Geht es auch ohne Datentypen?

17 Aufgabe

Was bedeutet eigentlich "Gleitpunktzahl einfacher oder doppelter Genauigkeit"? Wie wird solch eine Zahl im Rechner dargestellt? Wie wird eine Integer-Zahl dargestellt?

18 Aufgabe

Wie kann das 'A' eigentlich auf einem Computer dargestellt werden, der doch nur 0 und 1 kennt? Was wird im Rechner abgelegt für die Integer-Zahl 1 und für das Zeichen '1'.

19 Aufgabe

Ändern Sie das Ausgangsprogramm so ab, dass es auch **int** und **float** sowie einzelne **char** ausgeben kann. Was müssen Sie dafür tun?

20 Aufgabe

Bringen Sie das folgende Programm zum Laufen:

```
#include <stdio.h>

int main() { int a, b; double d; char ch;

  i = 4; a = 5.; b = 3; d = 4.5; ch = "A";

  printf("%d\n", a);
  printf("%f\n", b)
  printf("%5.2f\n", d);
  printf("%c\n, ch);
```

```
␣␣␣␣␣␣return(0);␣}
```

Welche Bedeutung hat die Format-Anweisung "5.2f"? Recherchieren Sie. Gibt es weitere Format-Anweisungen?

21 Aufgabe

Schreiben Sie ein C++-Programm, das folgende Ausgaben erzeugt:

1. eine Integer in dezimaler Schreibweise
2. eine Integer in hexadezimaler Form
3. eine Float in normaler Punkt-Darstellung
4. eine Float in wissenschaftlicher Darstellung, d.h. als $a * 10^b$
5. eine Zeichenkette

22 Aufgabe

Wie hängen die Begriffe **Anweisung** und **Ausdruck** mit den Begriffen **r-value** und **l-value** zusammen?

23 Aufgabe

Seien $f1$, $f2$, $f3$ **float** Variablen. Schreiben Sie die notwendige Code-Strecke, um $f2$ das doppelte von $f1$ und $f3$ die Hälfte von $f1$ zuzuweisen.

24 Aufgabe

Schreiben Sie ein Programm, das zwei Integer-Zahlen einliest und sowohl die Summe als auch das Produkt ausgibt. Das gleiche soll auch für float-Zahlen gebaut werden.

25 Aufgabe

Schreiben Sie ein Programm, dass für 10 verschiedene DM-Werte von 1 bis 5 DM die entsprechenden Euro-Werte errechnet und ausgibt. Woher können die DM-Werte kommen? Implementieren Sie zwei verschiedene Möglichkeiten. Welche Datentypen verwenden Sie?

26 Aufgabe

Schreiben Sie ein Programm, das für drei verschiedene Radien zwischen 2 und 3 jeweils die Oberfläche der Kugel und das Volumen berechnet und ausgibt. Die Radien können fest kodiert sein.

27 Aufgabe

Welche Darstellung haben die folgenden Ausdrücke in C:

$$(a + b)/(c - d) - de$$

und

$$\sqrt{3}(a + b)$$

28 Aufgabe

Was passiert jeweils bei den folgenden Zeilen. Welche Werte haben jeweils i, k, l. Erklären Sie.

```
int i, k, l; i = 42;

k = i++; l = ++i;
```

29 Aufgabe

Welche Ergebnisse haben die folgenden beiden Codezeilen, wenn zunächst $i = 4$ ist:

1. `(i++)++;`
2. `i++ * i`

30 Aufgabe

Wieso ist manchmal `i++;` schneller als `i = i + 1;`?

31 Aufgabe

Welchen Sinn machen die folgenden Ausdrücke:

```
int a, b, d;
a & 0707;
b | 0xC;
a = a ^ a;
b = b << 3;
d = d & ~07;
```

32 Aufgabe

Für welchen Wert der Integer-Variablen x ist der folgende Ausdruck wahr: `((x >= 10) && (x <= 200)) || (x == 5) && (x==6)`

33 Aufgabe

Betrachten Sie den folgenden Ausdruck: `(x < 0) || ((++x > 5) && (x < 7))`. Welchen Wert hat x nach Auswertung dieses Ausdrucks, wenn anfangs `x = -1;` war?

34 Aufgabe

Welche Werte haben die einzelnen Konstanten in der folgenden Aufzählung? Ist das immer so? Erklären Sie.

```
enum hugo {AA, BB = 3, CC='c', DD};
```

35 Aufgabe

Was passiert, wenn Sie eine Variable als `const` zu deklarieren und dann in Ihrem Programm verändern? Probieren Sie es aus!

36 Aufgabe

Welche Operatoren kennen Sie **in C**, die mit Zeichenketten arbeiten? Was machen Sie z.B., wenn Sie in einem **C-Programm** zwei Zeichenketten aneinander hängen wollen und was, wenn Sie Teile aus einer Zeichenkette verwenden wollen?

37 Aufgabe

Ändern Sie das Programm zur Umrechnung von DM in Euro so ab, dass es die umzuwandelnden Werte einliest und nur im Falle eines Wertes größer als 1000DM auch umwandelt. Sie wollen sich doch nicht mit Peanuts abgeben, oder?

38 Aufgabe

Berechnen Sie für einen eingegebenen Radius (Float) die Oberfläche und das Volumen einer Kugel mit diesem Radius. Beachten Sie, dass dies nur für positive Radien definiert ist.

39 Aufgabe

Wie sehen die Fluß-Diagramme für die ein-armige und zwei-armige **if**-Anweisung aus? Erklären Sie.

40 Aufgabe

Schreiben Sie ein Programm, das zwei Integer einliest und dafür das Maximum bestimmt. Verwenden Sie dafür

1. nur die ein-armige If-Anweisung
2. die zwei-armige `if-else` Anweisung

41 Aufgabe

Betrachten Sie folgendes Code-Segment:

```
if (a > 5)
   if (a < 10)
      b = c;
else
   cout << "a kleiner 5" << endl;
```

Was soll diese Code-Strecke ausdrücken? Was passiert hier tatsächlich? Haben Sie Verbesserungsvorschläge?

42 Aufgabe

Sie wollen einen Wert a einsortieren in die 5er-Intervalle von 5 aufwärts. Sie codieren dazu:

```
if (a > 5)
    cout << ">5" << endl;
  else if (a > 10)
    cout << ">10" << endl;
  else if (a > 15)
    cout << ">15" << endl;
  else
    cout << "a kleiner 5" << endl;
```

Was ist dazu zu sagen? Haben Sie Verbesserungsvorschläge?

43 Aufgabe

Schreiben Sie ein Programm, das 5 Buchstaben einliest und je nach Buchstabe unterschiedliche Zeichenketten ausgibt. Das Programm soll 3 von Ihnen gwählte Zeichen explizit verarbeiten können und den Rest durch einen gemeinsamen String beantworten.
Verwenden Sie hierfür einmal die `if-else`-Form und dann die `switch`-Anweisung. Welche ist hier sinnvoller?

44 Aufgabe

Schreiben Sie ein Programm, das einen Integer-Wert einliest und bei gerader Zahl das Quadrat ausgibt, bei ungerader die Wurzel. Verwenden Sie die

switch-Anweisung. Das Programm soll nur die Zahlen zwischen 1 und 14 akzeptieren. Was machen Sie bei einer fehlerhaften Eingabe?

45 Aufgabe

Welche Werte haben a und b nach dem Durchlauf der folgenden Schleifen. Die Initial-Werte seien $a = 5$ und $b = 0$.

```
while (a >= b) if (b==0) a--;
while (a >= b) {a=a-b; if (b==0) b++;}
```

46 Aufgabe

Schreiben Sie ein Programm, das Zeichen von der Eingabe liest. Das Programm soll auf die Eingabe eines “s” und eines “S” sich beenden. Andere Konsonanten sind als Fehler abzulehnen. Das letzte eingelesene Zeichen soll jeweils ausgegeben werden.

47 Aufgabe

Schreiben Sie ein Programm, das nur Zahlen von der Eingabe liest und die Summe ausgibt. Das Programm soll enden, wenn eine Null eingeben wird.

48 Aufgabe

Macht die folgende Code-Strecke einen Sinn und wenn ja, welchen?

```
do {
  /* do something important */
} while (0);
```

49 Aufgabe

Wie sieht das Fluß-Diagramm der **for**-Schleife aus?

50 Aufgabe

Was passiert bei folgender **for**-Schleife?

```
for (i = 0; i < 10; i++) i = i - 1;
```

51 Aufgabe

Schreiben Sie ein Programm, das die Summe der ersten hundert Integer-Werte ausgibt.

52 Aufgabe

Schreiben Sie ein Programm, das den größten gemeinsamen Teiler (ggT) zweier Integer-Zahlen u und v berechnet. Verwenden Sie dabei die Tatsache, daß, wenn u größer als v ist, der ggT durch den ggT von v und $u - v$ gegeben ist.

53 Aufgabe

Schreiben Sie ein Programm, das die Euler'sche Zahl e bis zu einer vorgegebenen Genauigkeit berechnet. Verwenden Sie dabei die Tatsache, dass gilt

$$e = \lim_{n \to \infty} a_n$$

mit

$$a_n = \sum_{i=0}^{n} \frac{1}{n!}$$

Dabei ist $n! = n * (n-1)! = 1 * 2 * ... * n$ die Fakultät von n.

54 Aufgabe

Schreiben Sie ein Programm, das - mittels **getchar()** - Zeichen aus der Eingabe liest und dabei zählt wieviel Ziffern, Zwischenräume und andere Zeichen dabei vorkamen. Dieses Ergebnis soll ausgegeben werden.

55 Aufgabe

Was ist der Unterschied zwischen den beiden folgenden Code-Strecken:

```
for (i = 0; i < 10; i ++) {
  if ( i % 2 == 0)
     continue;
  printf("%d: Hier\n", i);
}
```

und

```
for (i = 0; i < 10; i ++) {
  if ( i % 2 == 0)
     break;
  printf("%d: Hier\n", i);
}
```

56 Aufgabe

Was geschieht bei folgender Code-Strecke:

```
for (i = 0; i < 10; i++) {
  switch (i) {
    case 1: case 2: case 3:
        continue;
    case 4:
        break;
    default:
        break;
  }
  printf("%d: neuer Durchlauf\n", i);
}
```

57 Aufgabe

In der Vorlesung haben Sie die Struktur der Deklaration von Funktionen kennengelernt. Dabei sahen Sie, dass die Argument-Liste aus Paaren (Typ,Name) besteht.
In manchen Fällen kann jedoch der Name des Arguments weggelassen werden. Wann? Wieso? Begründen Sie.

58 Aufgabe

Wie sieht die Definition der kleinst-möglichen Funktion aus?

59 Aufgabe

Was passiert bei folgendem Programm und warum? Was sollte geschehen?

```
#include <stdio.h>

void myfunction(int a) {
  printf("Der Wert von a = %d", a);
}

int main() {
  int a = 42;
  myfun(a);
  return(0);
}
```

60 Aufgabe

Schreiben Sie die Deklarationen für folgende Funktionen:

1. Rückgabe von **float**, Eingabe von **int**, **int**, **char**
2. Rückgabe keines Wertes, Eingabe von **float**, **int**
3. Rückgabe von **int**, keine Eingabe.

Muß die Anzahl der Argumente immer bekannt sein? Was meinen Sie?

61 Aufgabe

Was ist eigentlich das allgegenwärtige

```
int main() {
  ...
  return(0);
}
```

62 Aufgabe

Was passiert in C bei folgender Code-Strecke:

```
void f (int a) {
  a++;
}
...
int c = 4;
f(c);
printf("Der Wert von c: %d\n", c);
```

Was bedeutet das für die Veränderung von Werte in einer Funktion? Womit arbeiten Sie im Falle des **call-by-value** also faktisch? Was geschähe im Falle eines **call-by-reference**?

63 Aufgabe

Schreiben Sie ein modulares Programm, das bei Eingabe dreier Integer das Maximum dieser Zahlen ermittelt und ausgibt.

64 Aufgabe

Schreiben Sie das ggT-Programm modular, d.h. verwenden Sie Funktionen.

65 Aufgabe

Teilen Sie das Programm zur Bestimmung des Maximums dreier Zahlen in mehrere Dateien auf. Welche sind hier sinnvoll? Wie erzeugen Sie daraus ein korrektes und funktionsfähiges Programm?

66 Aufgabe

Kann man in C Funktionen auch innerhalb anderer Funktionen definieren? Wie unterscheidet sich das von Variablen?

67 Aufgabe

Was passiert bei folgender Code-Strecke? Welchen Wert hat *var* nach Aufruf der Funktion. Erklären Sie.

```
int var = 5;
...
void func() {
  int var;
  var++;
  return var;
}
```

68 Aufgabe

Schreiben Sie ein Programm, das eine Funktion beinhaltet, die ausgibt, wie oft sie aufgerufen wurde. Verwenden Sie dazu keine globale Variable.

69 Aufgabe

Betrachten Sie folgende Aufruf-Hierarchie:

```
int f3(int a) {
  return a;
}

int f2 (int a, int b) {
  f3(a);
}

int f1(int a, int b, int c) {
  f2(a,b);
}
```

Wie sieht der Stack graphisch für diese Aufruf-Folge aus, wenn gerade die Funktion **f3** aktiv ist.

70 Aufgabe

Berechnen Sie die Fakultät einer eingegebenen Integer-Zahl durch eine Rekursion. Die Fakultät $f(n) = n!$ eines ganzzahligen Wertes ist definiert durch

$$f(n) = n * f(n-1) f(0) = 1;$$

71 Aufgabe

Schreiben Sie das Programm zur Bestimmung des größten gemeinsamen Teilers auf eine rekursive Form um und modularisieren Sie es in Header- und Implementierungsdateien.

72 Aufgabe

In der Vorlesung wurde behauptet, dass Rekursionen mitunter langsamer sind als Iterationen. Können Sie dafür eine Erklärung finden?

73 Aufgabe

Erst Datentypen geben den "rohen" Bitfolgen im Hauptspeicher des Rechners eine Bedeutung. Was ergibt z.B. die Bitfolge 100001, wenn man sie als Integer bzw. als Zeichen interpretiert?

74 Aufgabe

Welche Aktionen muß der Compiler für die folgenden Operationen durchführen?

```
int i;
long l;
float f;
double d;

char c;

i += l;
l += i;

l = i*f;
```

```
  f = l*f;

  d = f *d;

  c = i;
  i = c;
```

75 Aufgabe

Warum kann ein Einsatz von **typedef** sinnvoll sein?

76 Aufgabe

Deklarieren Sie die folgenden Felder:

1. ein Feld von 10 Integern
2. ein Feld von 20 Charactern
3. ein Feld von 6 Feldern aus je 4 Integer-Elementen

und greifen Sie auf jedes fünfte der vorhandenen Elemente zu.

77 Aufgabe

Erklären Sie mit eigenen Worten zeilenweise das Verhalten des Programms

```
#include <stdio.h>

int main() {
  char in[10] = "Hallo Welt";

  printf("%s\n", in);

  return(0);
}
```

78 Aufgabe

Wieso wird zwischen dem Zeichen 'c' und der Zeichenkette "c" unterschieden? Erklären Sie damit die Äquivalenz, die in der Vorlesung behauptet wurde:

```
char astr[] = "Hallo\n";  /* äquivalent */
char bstr[] = {'H', 'a', 'l', 'l', 'o', '\n', '\0'};
```

79 Aufgabe

Das Skalarprodukt zweier Vektoren x und y des R^n kann in Komponenten-Schreibweise geschrieben werden als

$$x \cdot y = \sum_{i=0}^{n} x_i y_i$$

Schreiben Sie ein Programm, dass für zwei Vektoren des R^{11} das Skalarprodukt berechnet. Verwenden Sie dazu Felder und eine Berechnungsfunktion.

80 Aufgabe

Schreiben Sie ein Programm, dass aus einer Reihe von Zahlen (integer oder float) die Größte heraus sucht. Verwenden Sie dazu ein Feld gegebener Größe, das die Zahlen speichert und eine Funktion, in der die Bestimmung stattfindet.

81 Aufgabe

Schreiben Sie ein Programm, das die Länge einer Zeichenkette ausgibt. Die Länge soll dabei **nicht** der verantwortlichen Funktion angegeben werden

82 Aufgabe

Schreiben Sie ein Programm, das eine Zeichenkette bis zu einer maximalen Länge einliest und in umgekehrter Reihenfolge wieder ausgibt. Verwenden Sie dazu Funktionen und Felder.

83 Aufgabe

Schreiben Sie ein Programm, das eine einzugebene Folge von Integer-Zahlen der Reihe nach sortiert und wieder ausgibt. Sie können dazu den einfachen Bubble-Sort verwenden. Schauen Sie ggf. in der Literatur nach.

84 Aufgabe

Felder als Funktionsargumente benötigen etwas Aufmerksamkeit, da die Länge nicht durch den Namen des Arguments gekennzeichnet ist. Wie steht es denn bei der Rückgabe eines Feldes? Geht das überhaupt, was haben Sie dabei zu beachten?

85 Aufgabe

Deklarieren Sie die folgenden **structs**:

1. eine struct aus einer Integer-Variable, einem Integer-Feld der Länge 2 und einem Float
2. eine struct aus einer Double, einem Character-String der Länge 20 und der struct aus Punkt 1
3. ein Feld der Länge 10 aus structs von Punkt 1
4. eine struct aus einem float, einem int und einem Feld von 4 structs des Typs zwei

Können Strukturen überhaupt Strukturen als Komponenten besitzen? Wieso wird hierbei eigentlich immer nur von der Deklaration gesprochen?

86 Aufgabe

Ist der folgende Code korrekt? Begründen Sie.

```
int aint;
struct bla {
  int aint;
  float fl;
};
```

87 Aufgabe

Schreiben Sie ein Programm, das Bücher-Bestellungen entgegennehmen kann. Ein Buch soll dabei durch den Titel, den Autor, der ISBN und der Anzahl der Seiten beschrieben werden. Diese Angaben sollen solange der Eingabe des Anwenders entnommen werden, bis der Anwender ein Ende wünscht.

88 Aufgabe

Vergleichen Sie Felder und Strukturen bzgl. der folgenden Eigenschaften:

1. neuer Datentyp
2. explizite Initialisierung
3. Vergleichs-Operator (==)
4. Zuweisung, d.h. $a = b$
5. Funktionsargument und Rückgabewert

89 Aufgabe

Wie groß sind die jeweiligen Datentypen? Prüfen Sie Ihre Antwort in einem Programm nach.

```
struct dop{
  int a;
  int b[2];
  float f;
};

union au {
  int i;
  double d;
};
```

90 Aufgabe

Betrachten Sie die folgende Verwendung einer **union**.

```
union ttt {
  int i;
  float f;
};
union ttt uvar;
...
int a = uvar.i;
```

Was ist dabei problematisch? Wie kann man das Problem lösen?

91 Aufgabe

Skizzieren Sie den generellen Aufbau eines C-Programms. Wie werden die verschiedenen Bestandteile des Programms auf unterschiedliche Dateien verteilt? Was muß dabei alles beachtet werden?

```
#include <stdio.h>

int main() {

  return(0);
}
```