

Angewandte Informatik

Übungsaufgaben – 21. Januar 2016

Aufgabe 1: Objektorientierte Tier-Datenbank

(Klausuraufgabe aus dem Wintersemester 2013/14)

```
#include <stdio.h>

#define ANIMAL 0
#define WITH_WINGS 1
#define WITH_LEGS 2

typedef struct animal
{
    int type;
    char *name;
} animal;

typedef struct with_wings
{
    int wings;
} with_wings;

typedef struct with_legs
{
    int legs;
} with_legs;

int main (void)
{
    animal *a[2];

    animal duck;
    a[0] = &duck;
    a[0]—>type = WITH_WINGS;
    a[0]—>name = "duck";
    a[0]—>wings = 2;  ← ((with_wings *) a[0])—>wings = 2;

    animal cow;
    a[1] = &cow;
    a[1]—>type = WITH_LEGS;
    a[1]—>name = "cow";
    a[1]—>legs = 4;  ← ((with_legs *) a[1])—>legs = 4;

    for (int i = 0; i < 2; i++)
        if (a[i]—>type == WITH_LEGS)
            printf ("A_%s_has_%d_legs.\n", a[i]—>name,
                    ((with_legs *) a[i])—>legs);
        else if (a[i]—>type == WITH_WINGS)
            printf ("A_%s_has_%d_wings.\n", a[i]—>name,
                    ((with_wings *) a[i])—>wings);
        else
            printf ("Error_in_animal:_%s\n", a[i]—>name);

    return 0;
}
```

Das oben in Blau dargestellte Programm ([aufgabe-1a.c](#)) soll Daten von Tieren verwalten.

Beim Compilieren erscheinen die folgende Fehlermeldungen:

```
$ gcc -std=c99 -Wall -O aufgabe-1a.c -o aufgabe-1a
aufgabe-1a.c: In function 'main':
aufgabe-1a.c:31: error: 'animal' has no member named 'wings'
aufgabe-1a.c:37: error: 'animal' has no member named 'legs'
```

Der Programmierer nimmt die oben in Rot dargestellten Ersetzungen vor (Datei [aufgabe-1b.c](#)).

Daraufhin gelingt das Compilieren, und die Ausgabe des Programms lautet:

```
$ gcc -std=c99 -Wall -O aufgabe-1b.c -o aufgabe-1b
$ ./aufgabe-1b
A duck has 2 legs.
Error in animal: cow
```

- Erklären Sie die o. a. Compiler-Fehlermeldungen. (2 Punkte)
- Wieso verschwinden die Fehlermeldungen nach den o. a. Ersetzungen? (3 Punkte)
- Erklären Sie die Ausgabe des Programms. (5 Punkte)
- Beschreiben Sie – in Worten und/oder als C-Quelltext – einen Weg, das Programm so zu berichtigen, daß es die Eingabedaten ("A duck has 2 wings. A cow has 4 legs.") korrekt speichert und ausgibt. (4 Punkte)

Aufgabe 2: Objektorientierte Programmierung mit dem C-Datentyp *union*

(Klausuraufgabe aus dem Wintersemester 2014/15)

In Wintersemester 2014/15 wurde der Datentyp *union* in der Vorlesung *nicht* behandelt.)

Aus der Vorlesung ist der Verbund-Datentyp *struct* bekannt.

Die Programmiersprache C kennt noch einen weiteren Verbund-Datentyp *union*:

```
typedef union
{
    int number;
    char *name;
    uint8_t bytes[4];
} data;
```

Von der Syntax her entspricht die *union* genau dem *struct*. Insbesondere lassen sich die Datenfelder über einen Punkt ansprechen, bei Zeigern auch mit *—>*. Der Unterschied zum *struct* besteht darin, daß sich in einer *union* die Datenfelder *dieselben Speicherzellen teilen*. Alle Datenfelder einer *union* haben *dieselbe Speicheradresse*. (In einem *struct* beginnt das nächste Datenfeld immer dort, wo das vorherige aufhört.)

Bei Zuweisung eines Wertes an *ein* Datenfeld einer *union* ändern sich daher die Werte *aller* Datenfelder. Dies kann man nutzen, um Speicher zu sparen (wenn man immer nur eines der Datenfelder auf einmal nutzen möchte) oder um „verwandte“ Datentypen zu konstruieren (im Sinne der objektorientierten Programmierung) oder um die Byte-Muster von Variablen zu untersuchen.

- (a) Was bewirkt das folgende Programm ([aufgabe-2a.c](#)), wenn es auf einem LittleEndian-Rechner ausgeführt wird, und warum? (4 Punkte)

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

typedef union
{
    uint32_t number;
    char *name;
    uint8_t bytes[4];
} data;

int main (void)
{
    data x;
    x.number = 303108111;
    for (int i = 0; i < 4; i++)
        printf ("%d_", x.bytes[i]);
    printf ("\n");
    printf ("%s\n", x.name);
    return 0;
}
```

- (b) Was würde dasselbe Programm auf einem BigEndian-Rechner ausgeben? (1 Punkt)

Wir betrachten nun das folgende Programm (aufgabe-2c.c):

```
#include <stdio.h>

#define POINT 0
#define CIRCLE 1
#define TEXT 2

typedef union
{
    int radius;
    char *text;
} extra_data;

typedef struct graphics_object
{
    int type;
    void (*draw) (struct graphics_object *this);
    int x, y;
    extra_data data;
} graphics_object;

void draw_point (struct graphics_object *this)
{
    printf ("point_at_(%d,%d)\n", this->x, this->y);
}

void draw_circle (struct graphics_object *this)
{
    printf ("circle_at_(%d,%d)_with_radius_%d\n",
        this->x, this->y, this->data.radius);
}

void draw_text (struct graphics_object *this)
{
    printf ("text_at_(%d,%d):_\"%s\"\n",
        this->x, this->y, this->data.text);
}

int main (void)
{
    graphics_object a_point = { POINT, draw_point, 35, 17 };
    graphics_object a_circle = { CIRCLE, draw_circle, 20, 30 };
    a_circle.data.radius = 12;
    graphics_object some_text = { TEXT, draw_text, 42, 23 };
    some_text.data.text = "Hello,_world!";
    graphics_object *g[3] = { &a_point, &a_circle, &some_text };
    for (int i = 0; i < 3; i++)
        g[i]->draw (g[i]);
    return 0;
}
```

Dieses Programm verwaltet verschiedenartige Grafikobjekte in einem Array `graphics_object *g[3]`. Anstatt tatsächlich zu zeichnen, gibt es der Einfachheit halber als Text aus, was ggf. zu sehen wäre.

Der Datentyp `graphics_object` ist ein **struct**, in dem eins der Datenfelder eine **union** (mit weiteren Datenfeldern) ist.

- (c) Beschreiben Sie (in Worten, evtl. mit Skizze) die in diesem Programm realisierte Objekt-Hierarchie. Worum handelt es sich insbesondere bei `void (*draw) (struct graphics_object *this)`? (3 Punkte)
- (d) Was passiert, wenn man im Hauptprogramm `main()` die an `a_circle` übergebene Funktion `draw_circle` durch `draw_text` ersetzt (aufgabe-2d.c) und warum? (2 Punkte)
- (e) Erweitern Sie das Programm so, daß es einen weiteren Grafikobjektyp `SQUARE` kennt, bei dem man eine Kantenlänge `a` angibt und das beim „Zeichnen“ den Text „square at (x,y) with edge length a“ (mit den korrekten Zahlen anstelle von `x`, `y` und `a`) ausgibt. (5 Punkte)

Abgabe auf Datenträger ist in der Klausur erwünscht, aber nicht zwingend.