

Angewandte Informatik – Klausur – 30. Januar 2015

Prof. Dr. Peter Gerwinski, Wintersemester 2014/15

Name:	
Matrikel-Nr.:	

Zeit: 120 Minuten

Zulässige Hilfsmittel:

- Schreibgerät
- Beliebige Unterlagen in Papierform und/oder auf Datenträgern
- Elektronische Rechner (Notebook, Taschenrechner o. ä.)
ohne Zugang zu Datennetzen jeglicher Art

Nur die o. a. zulässigen Hilfsmittel dürfen sich während der Klausur im Arbeitsbereich befinden. WLAN-, Bluetooth- und sonstige Funkeinheiten von Notebooks o. ä. sind auszuschalten; ggf. dafür vorhandene physische Schalter sind zu benutzen. Mobiltelefone, Geräte mit mobilem Internet-Zugang u. ä. sind auszuschalten und in der Tasche zu verstauen.

Die reguläre Maximalpunktzahl beträgt 42 Punkte.

Bei besonderen Leistungen sind Zusatzpunkte möglich.

Mit 20 erreichten Punkten gilt die Klausur als bestanden.

Das Passwort für den Zugriff auf die Beispielpprogramme lautet: **nE6LxcNG3sXA**

Aufgabe 1: Arrays mit Zahlen

Wir betrachten das folgende Programm ([aufgabe-1.c](#)):

```
#include <stdio.h>

void f (int *s0, int *s1)
{
    while (*s0 >= 0)
    {
        int *s = s1;
        while (*s >= 0)
            if (*s0 == *s++)
                printf ("%d_", *s0);
        s0++;
    }
    printf ("\n");
}

int main (void)
{
    int a[] = { 10, 4, 3, 7, 12, 0, 1, -1 };
    int b[] = { 7, 14, 0, 8, 9, 22, 10, -1 };
    f (a, b);
    return 0;
}
```

(a) Was bewirkt die Funktion `f` und warum? (4 Punkte)

(b) Von welcher Ordnung (Landau-Symbol) ist die Funktion und warum?

Wir beziehen uns hierbei auf die Anzahl der Vergleiche in Abhängigkeit von der Länge der Eingabedaten **s0** und **s1**. Für die Rechnung dürfen Sie beide Längen mit n gleichsetzen, obwohl sie normalerweise nicht gleich sind. (2 Punkte)

- (e) Beschreiben Sie – in Worten und/oder als C-Quelltext –, wie sich die Funktion `f` effizienter gestalten läßt, wenn man die ihr übergebenen Arrays `s0` und `s1` als sortiert voraussetzt. (5 Punkte)

- (f) Von welcher Ordnung (Landau-Symbol) ist Ihre effizientere Version der Funktion und warum? (2 Punkte)

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

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;
}
```

A full page of blank graph paper with a uniform grid of small squares. The grid consists of 20 columns and 20 rows, creating a total of 400 squares. The lines are thin and gray, set against a white background. There are no margins or additional markings on the page.

- (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);
}

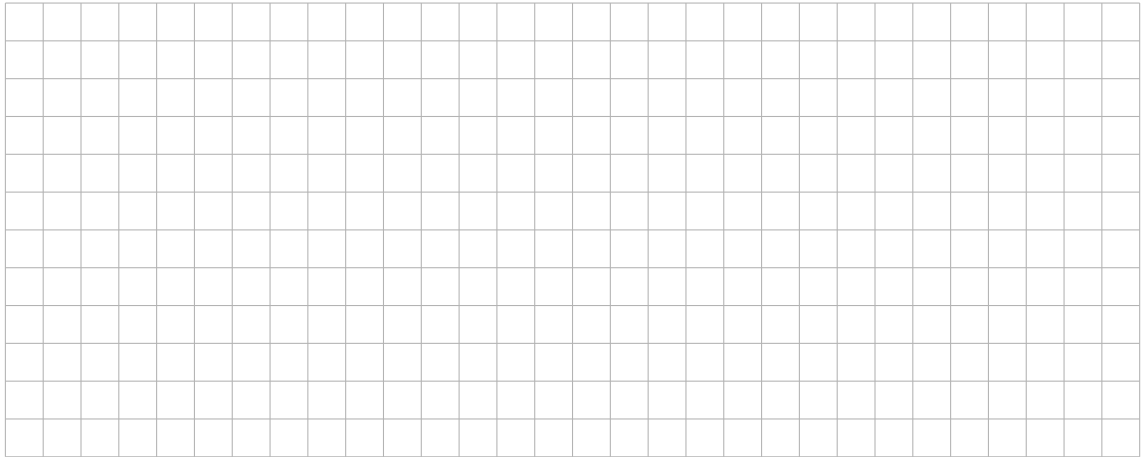
int main (void)
{
    graphics_object a_point = { POINT, draw_point, 35, 17 };
    graphics_object a_circle = { CIRCLE, draw_circle, 20, 30 };
    a_circle.data.radius = 12;
    graphics_object some_text = { TEXT, draw_text, 42, 23 };
    some_text.data.text = "Hello,_world!";
    graphics_object *g[3]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

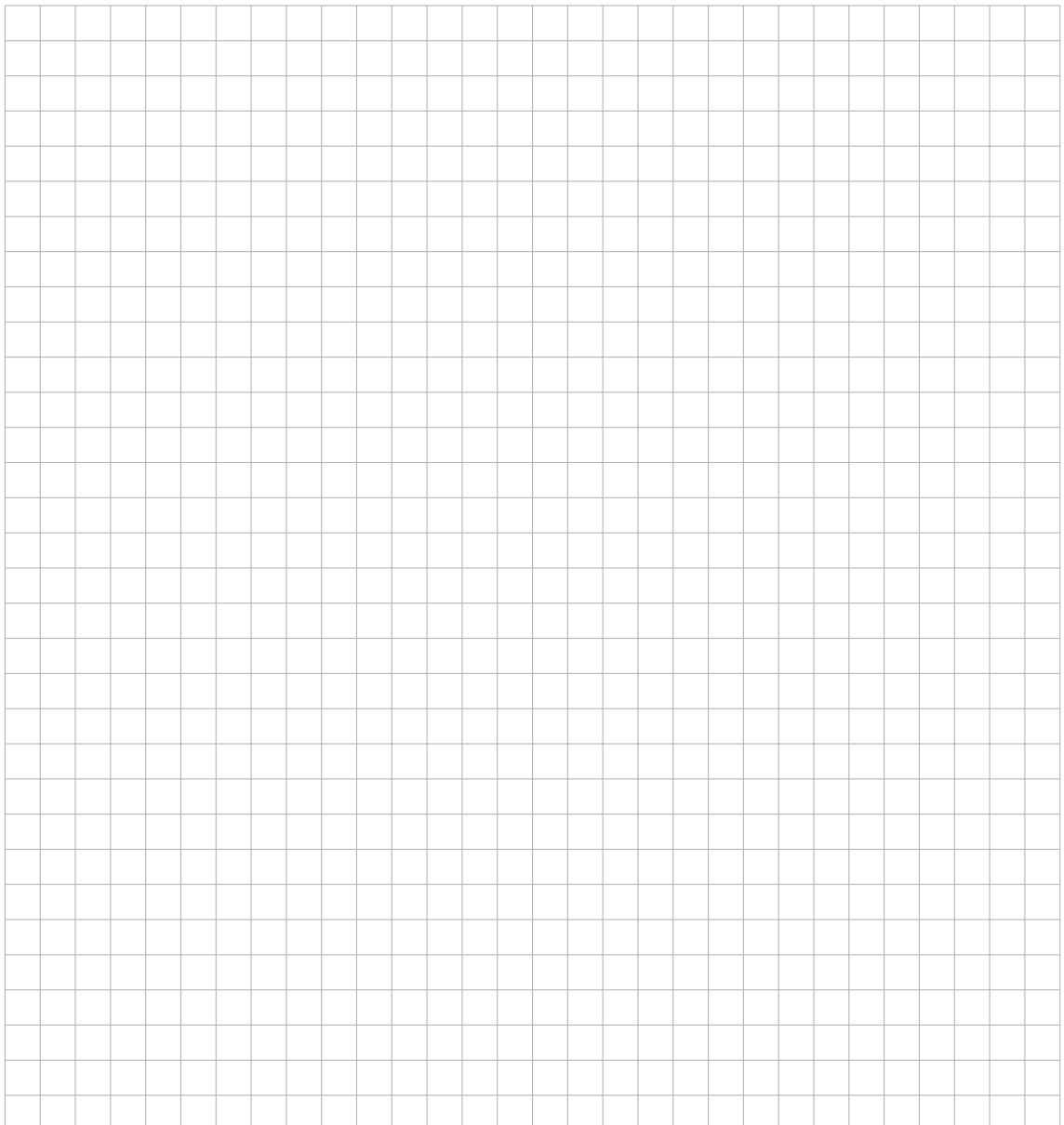
- (c) Beschreiben Sie (in Worten, evtl. mit Skizze) die in diesem Programm realisierte Objekt-Hierarchie. Worum handelt es sich insbesondere bei **void (*draw) (struct graphics_object *this)**? (3 Punkte)

- (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 Grafikobjekttyp `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)

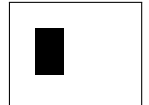
Abgabe auf Datenträger ist erwünscht, aber nicht zwingend. Raum für Notizen:



Aufgabe 3: PBM-Grafikbibliothek

PBM-Grafikdateien sind ähnlich aufgebaut wie PPM-Grafikdateien (siehe die Beispielpprogramme [ppm-1.c](#) bis [ppm-8.c](#) aus der Vorlesung). Die Unterschiede lauten:

- Die Kennung lautet „P4“ anstelle von „P6“.
- Es gibt keine beliebigen Farben mit jeweils einem Byte Rot-, Grün- und Blauanteil, sondern nur die Farben Schwarz (1) und Weiß (0).
- Die Angabe der Farbtiefe entfällt. Vor den eigentlichen Bilddaten kommen nur die Kennung, Breite und Höhe des Bildes.
- Jeder Bildpunkt belegt 1 Bit; es werden also 8 Bildpunkte in jedem Byte gespeichert (im Gegensatz zu 1 Bildpunkt pro 3 Bytes in PPM-Dateien). Die Anordnung der Bits ist *MSB First*, also dieselbe wie im VGA-Grafikspeicher.
- Beispiel-Bild ([aufgabe-3.pbm](#), entspricht [aufgabe-3.png](#)): ein Rechteck von 5×8 schwarzen Pixeln auf weißem Grund. Die linke, obere Ecke des Rechtecks liegt bei den Koordinaten (3, 3); die Gesamtgröße des Bildes ist 20×15 Pixel.



Schreiben Sie die unten beschriebenen Funktionen zur Erstellung von PBM-Grafikdateien.

- **void pbm_init (int width, int height)**
Einen virtuellen Zeichenbereich mit der (variablen) Breite **width** und Höhe **height** erzeugen und auf die Farbe Weiß initialisieren („Bildschirm löschen“).
- **void pbm_set_pixel (int x, int y)**
Im virtuellen Zeichenbereich den Punkt (Pixel) mit den Koordinaten (x, y) auf die Farbe Schwarz setzen.
- **void pbm_save (char *filename)**
Den Inhalt des virtuellen Zeichenbereichs unter dem angegebenen Dateinamen in einer PBM-Grafikdatei speichern.
- **void pbm_done (void)**
Den dynamisch belegten Speicherplatz wieder freigeben.
- Schreiben Sie ein Testprogramm, das die Funktionen nutzt, um eine PBM-Datei mit einem schwarzen Quadrat auf weißem Grund zu erzeugen.

Die Funktionen sollen möglichst robust sein, d. h. das Programm soll auch bei unsinnigen Parameterwerten nicht abstürzen, sondern eine Fehlermeldung ausgeben.

(10 Punkte)

Abgabe auf Datenträger ist erwünscht, aber nicht zwingend. Raum für Notizen: