

Angewandte Informatik – Klausur – 18. September 2014

Prof. Dr. Peter Gerwinski, Wintersemester 2013/14

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 per Hardware auszuschalten.
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.

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 s_0 und s_1 . Für die Rechnung dürfen Sie beide Längen mit n gleichsetzen, obwohl sie normalerweise nicht gleich sind. (1 Punkt)

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

- (f) Von welcher Ordnung (Landau-Symbol) ist Ihre effizientere Version der Funktion und warum? (1 Punkt)

Aufgabe 2: Objektorientiertes Grafik-Programm

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include "opengl-magic.h"

#define GRAPH_OBJECT 0
#define SPHERE 1
#define TEAPOT 2

typedef struct
{
    int type;
} graph_object;

typedef struct
{
    int type;
    double r;
    double x, y, z;
    double color_r, color_g, color_b;
} sphere;

typedef struct
{
    int type;
    double x, y, z;
    double angle_x, angle_y;
    double size;
    double color_r, color_g, color_b;
} teapot;

void draw_object (graph_object *g)
{
    if (g->type == SPHERE)
    {
        sphere *s = (sphere *) g;
        glPushMatrix ();
        glTranslatef (s->x, s->y, s->z);
        set_material_color (s->color_r, s->color_g, s->color_b);
        glutSolidSphere (s->r, 31, 10);
        glPopMatrix ();
    }
    else if (g->type == TEAPOT)
    {
        teapot *t = (teapot *) g;
        glPushMatrix ();
        glTranslatef (t->x, t->y, t->z);
        glRotatef (t->angle_x, 1, 0, 0);
        glRotatef (t->angle_y, 0, 1, 0);
        set_material_color (t->color_r, t->color_g, t->color_b);
        glutSolidTeapot (t->size);
        glPopMatrix ();
    }
}

void draw (void)
{
    teapot t = {
        type: TEAPOT,
        x: 0.3,
        y: -0.4,
        z: 0.2,
        angle_x: 0.0,
        angle_y: -30.0,
        size: 0.3,
        color_r: 1.0,
        color_g: 0.5,
        color_b: 0.0
    };
    glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    draw_object (t); ← draw_object ((graph_object *) &t);
    t.type = SPHERE;
    draw_object (t); ← draw_object ((graph_object *) &t);
    glFlush ();
    glutSwapBuffers ();
}

int main (int argc, char **argv)
{
    init_opengl (&argc, argv, "Graphics_Test");
    return 0;
}
```

Das oben in Blau dargestellte Programm ([aufgabe-2a.c](#)) soll 3d-Grafikobjekte zeichnen. (Die verwendeten Bibliotheken sind aus der Vorlesung bekannt.)

Beim Compilieren erscheinen die folgende Fehlermeldungen:

```
$ gcc -Wall -O aufgabe-2a.c -lGL -lGLU -lglut opengl-magic.c -o aufgabe-2a
aufgabe-2a.c: In function 'draw':
```

Der Programmierer nimmt die oben in Rot dargestellten Ersetzungen vor (Datei [aufgabe-2b.c](#)). Daraufhin gelingt das Compilieren. Beim Aufruf des Programms passiert allerdings nichts Erkennbares.

-
- A 3D rendering of a yellow teapot and a blue sphere on a black background. The teapot is in the lower-left foreground, and the sphere is in the upper-right background.

- (c) In Hinblick auf objektorientierte Programmierung weist das Programm diverse Design-Schwächen auf. Nennen Sie – in Worten und/oder als C-Quelltext – Möglichkeiten, das Programm in dieser Hinsicht zu verbessern. (4 Punkte)

Aufgabe 3: PBM-Grafikbibliothek

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

(PBM-Grafikdateien sind Ihnen sowohl aus der Lehrveranstaltung „Angewandte Informatik“ als auch aus der Lehrveranstaltung „Grundlagen Rechnerntechnik“ bekannt.)

- **void pbm_init (int width, int height)**
Einen virtuellen Zeichenbereich mit der (variablen) Breite **width** und Höhe **height** erzeugen.
- **void pbm_clear (void)**
Den virtuellen Zeichenbereich 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_clear_pixel (int x, int y)**
Im virtuellen Zeichenbereich den Punkt (Pixel) mit den Koordinaten (x, y) auf die Farbe Weiß setzen.
- **int pbm_get_pixel (int x, int y)**
Die Farbe des Punktes mit den Koordinaten (x, y) zurückliefern (0 = Weiß, 1 = Schwarz).
- **void pbm_fill (int x, int y)**
Einen schwarz umrandeten Bereich um den Punkt (x, y) herum mit der Farbe Schwarz ausfüllen.
- **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 Dreieck auf weißem Grund zu erzeugen.

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

(14 Punkte)

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

