Einführung in C++ – Übung 3
Testatgruppe A (Isaak)

Rasmus Diederichsen Simon Kern

27. Oktober 2014

Aufgabe 3.1 Pointerarithmetik

Listing 1: pointer_test.c

```
1  #include <stdio.h>
2  int main(void)
3  {
4     int feld[5];
5     int *p, *p1, *p2;
6     int i;
7
8     p = feld; /* a) Geht, pointer kann immer auf array zeigen */
9     /* feld = p; */ /* b) Geht nicht, da arrays nicht neu zugewiesen
           werden koennen */
10    p = &feld[3]; /* c) geht, p zeigt dann auf 4. element von feld
          */
11    feld[2] = p[5]; /* d) trivial */
12    p1 = p2 + i; /* e) pointer koennen vor- oder zurueckgerueckt
          werden */
13    p1 = i + p2; /* f) s.o. */
14    /* i = p1 * p2; */ /* g) geht nicht, adressen zu multipliziern,
          ist sinnlos */
15    i = p1 - p2; /* h) einzige erlaubte arithmetische operation auf
          2 pointern. Damit kann bestimmt werden, wie weit zwei dinge
          auseinander liegen*/
16    /* i = p1 + p2; */ /* i) geht nicht, ist sinnlos */
17
18    /* Addition ist kommutativ */
19    feld[i]; /* syntactic sugar fuer *(feld + i) */
20    i[feld]; /* syntactic sugar fuer *(i + feld) */
21    printf("huu\n");
22    return 0;
23 }
```

Zweidimensionale Arrays sind Pointer, die auf den Beginn eines Speicherbereichs zeigen, der Pointer zum entsprechenden Datentyp enthält. Diese wiederumg zeigen

dann auf einen eigenen Block (der irgendwo liegen kann) der dann die primitiven Daten enthält.

Aufgabe 3.2 Einarbeitung in `cmake`

`cmake` vereinfacht gegenüber `make` vor allem das Einbinden von Bibliotheken, die man in normalen Makefiles oft manuell suchen und verlinken muss. `cmake` kommt mit Skripten, die zum Auffinden vieler Bibliotheken für verschiedene Systeme dienen, und so den Kompilationsprozess relativ plattformunabhängig gestalten. Grundlegende Funktionalitäten sind die Eingabe von In- und Outputverzeichnissen, das Spezifizieren von Build-Targets und den dazugehörigen Bibliotheken, sowie das Kompilieren zu einer Bibliotheksdatei. Es ist auch einfach möglich, nicht-standard-artige Verezeichnisse zum Durchsuchen an `cmake` zu übergeben.

Der Unterschied zwischen In-Source und Out-of-Source-Builds ist, dass für die letzteren sämtliche beim Build entstehenden Dateien an einen Ort außerhalb des Quellverzeichnisses abgelegt werden.

Aufgabe 3.3 Hauptfenster mit `glut`

Listing 2: mainwindow.h

```
1  #ifndef MAINWINDOW_MLTZT4GG
2
3  #define MAINWINDOW_MLTZT4GG
4
5  /**
6   * @brief GLUT callback for rendering the window content.
7   */
8  void render(void);
9
10 /**
11  * @brief GLUT callback invoked upon changes of window size.
12  * @param w The new width.
13  * @param h The new height.
14  */
15 void reshape(int w, int h);
16
17 /**
18  * @brief GLUT callback processing key presses when window has
        focus.
19  * @param key The key pressed.
20  * @param x The x coordinate of the mouse at the time of key press.
21  * @param y The y coordinate of the mouse at the time of key press.
22  */
23 void keyPressed(unsigned char key, int x, int y);
24
25 /**
26  * @brief GLUT callback invoked when mouse is clicked in window.
27  * @param button The mouse button clicked.
28  * @param x The x coordinate of the mouse at the time of key press.
29  * @param y The y coordinate of the mouse at the time of key press.
30  */
```

```
31 void mouseMoved(int button, int state, int x, int y);
32
33 #endif /* end of include guard: MAINWINDOW_MLTZT4GG */
```

Listing 3: mainwindow.c

```
1  #include "include/mainwindow.h"
2  #include <glut.h>
3  #include <stdio.h>
4
5  void render(void)
6  {
7     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
8
9     glBegin(GL_TRIANGLES);
10       glVertex3f(-0.5,-0.5,0.0);
11       glVertex3f(0.5,0.0,0.0);
12       glVertex3f(0.0,0.5,0.0);
13    glEnd();
14
15    glutSwapBuffers();
16
17 }
18
19 void reshape(int w, int h)
20 {
21    if(h == 0)
22       h = 1;
23    float ratio = 1.0* w / h;
24
25    glMatrixMode(GL_PROJECTION);
26
27    glLoadIdentity();
28
29    glViewport(0, 0, w, h);
30
31    gluPerspective(45,ratio,1,1000);
32
33    glMatrixMode(GL_MODELVIEW);
34
35 }
36
37 void keyPressed(unsigned char key, int x, int y)
38 {
39    printf("The␣key␣pressed␣was␣%c.␣Mouse␣coordinates␣are␣(%d,%d).\n
       ",
40          key, x, y);
41 }
42
43 void mouseMoved(int button, int state, int x, int y)
44 {
45    printf("(%d,%d)", x, y);
46    switch(button)
47    {
48       case GLUT_LEFT_BUTTON:
49          printf("␣+␣left␣button");
50          break;
51       case GLUT_RIGHT_BUTTON:
```

```
52            printf("␣+␣right␣button");
53            break;
54        case GLUT_MIDDLE_BUTTON:
55            printf("␣+␣middle␣button");
56            break;
57    }
58    printf("\n");
59 }
```

Listing 4: main.c

```
1  #include <glut.h>
2  #include "include/mainwindow.h"
3  int main(int argc, char *argv[])
4  {
5     glutInit(&argc, argv);
6     glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
7     glutInitWindowPosition(100,100);
8     glutInitWindowSize(320,320);
9     glutCreateWindow("test");
10
11    glutMouseFunc(mouseMoved);
12    glutKeyboardFunc(keyPressed);
13
14    glutDisplayFunc(render);
15    glutReshapeFunc(reshape);
16
17    glutMainLoop();
18
19    return 0;
20 }
```

```
1  project(Blatt_3 C)
2  set(CMAKE_C_COMPILER gcc) # why is this cc by default?
3  set(CMAKE_C_FLAGS "-Wall␣-Wstrict-prototypes␣-ansi␣-pedantic")
4
5  # ex 1
6  add_executable(pointer_test pointer_test.c)
7
8  # ex 2
9  find_package(GLUT REQUIRED)
10 find_package(OpenGL REQUIRED)
11
12 # header files
13 include_directories(${GLUT_INCLUDE_DIR})
14
15 add_executable(mainwindow main.c mainwindow.c)
16 target_link_libraries(mainwindow ${GLUT_LIBRARIES} ${
       OPENGL_LIBRARIES})
```