1 Allgemeines

1.1 Lernziele

Ziel der Übung ist es, die in der Vorlesung erworbenen Kenntnisse zu vertiefen und praktisch anzuwenden, sowie Sie zum Selbststudium anzuleiten. Außerdem sollen Sie erste Erfahrungen sammeln im gemeinsamen Erstellen von Software: Man einigt sich auf eine Schnittstelle (diese wird in der Regel von uns vorgegeben) und jeder Beteiligte löst unabhängig vom Anderen einen Teil der Aufgabe. Am Ende müssen alle Module zusammen kompilierbar sein und das Programm die gewünschte Funktionalität aufweisen.

Zur Software-Entwicklung gehört auch das systematische Testen und eine zielgerichtete Fehlersuche: Sind die einzelnen Komponenten fehlerhaft oder arbeiten sie nur nicht korrekt zusammen. Vielleicht sind sogar in den von uns bereitgestellten Programmteilen Fehler enthalten. Kenntnisse zum Testen und Debuggen von Programmen werden ebenfalls im Rahmen der Übungen erworben.

1.2 Organisatorisches

Vorbereitung Alle Aufgaben sind zu Hause vorzubereiten, d.h. die Vorlesungsinhalte müssen nachgearbeitet werden und verstanden sein. Die Vorbereitung dient dazu, dass Sie sich kritisch mit den in der Vorlesung behandelten Inhalten beschäftigen und dass Sie nicht behandelte Themen selbständig erarbeiten. Eine gute Mitarbeit in und an den Übungen ist hilfreich zum Verstehen der Vorlesungsinhalte und damit zum Bestehen der Klausur.

Motivation Im Laufe der Übung kann eine etwas umfangreichere Software entstehen. An jedem Übungstermin kann ein kleiner Teil der Software implementiert werden, sodass am Ende daraus ein echtes Produkt zusammengesetzt werden kann. Jeder Teil wird für sich getestet (Komponententest). Die Funktionalität wächst also im Laufe der Zeit. Man nennt dieses Vorgehen auch inkrementelle Software-Entwicklung. Dazu ist auf jedem Übungsblatt eine mit *Routenplanung* gekennzeichnete Aufgabe enthalten, die am Ende einen, man glaubt es kaum, Routenplaner ergibt. Darin soll eine Karte mittels eines Graphen modelliert werden. Die Knoten repräsentieren die Städte (oder Straßenkreuzungen) und die Kanten die Strecken dazwischen. Weitere Funktionalitäten wie das Finden der kürzesten Wegstrecke, effiziente Möglichkeiten der Integration von Wegänderungen bei Stau und Sperrungen, sowie Streckenplanung mit mehreren Zielpunkten können zusätzlich implementiert werden.

Programmierung Bei der Erstellung der Programme sollten Sie sich an folgenden Fragen orientieren:

- Erfüllt das Programm die in der Aufgabe beschriebene Funktionalität? Sind die Tests ausreichend?
- Ist das Programm lesbar und verständlich? Wurden logisch zusammen gehörende Teile in Funktionen/Methoden gekapselt? Sind die Namen von Variablen/Attributen und Funktionen/Methoden gut gewählt?
- Wird Speicherplatz oder Laufzeit verschwendet?
- Sind die Funktionen/Methoden zu lang, überfrachtet oder erfüllen mehr als eine Aufgabe?

Kommunikation im Forum Für alle inhaltlichen sowie organisatorischen Fragen bitten wir Sie, das Moodle-Forum zu nutzen. Stellen Sie Ihre Fragen bitte direkt in dem entsprechenden Forum. Über das Forum erhalten Sie zeitnah eine Antwort betreuender Lehrender oder Mitarbeiter/innen. Außerdem profitieren so auch die anderen Studierenden von Ihren Fragen. Alle nicht-persönlichen Anfragen, die dennoch per E-Mail eingehen, werden dort ebenfalls eingestellt.

2 C++ Coding Guidelines

Die folgenden Richtlinien wurden von Herb Sutter und Bjarne Stroustrup erarbeitet und sollen in der Übung beachtet werden. Sie finden diese Richtlinien im Web unter der folgenden URL: `https://github.com/isocpp/CppCoreGuidelines`

Diese Richtlinien sollen dafür sorgen, dass auch andere Personen Ihren Code lesen und verstehen können, außerdem sollen so Fehler beim Programmieren vermieden werden. Wir geben hier nur einen kleinen Auszug an, da wir nicht alle in den Richtlinien verwendeten Spracheigenschaften in der Vorlesung kennen lernen werden.

Beispiele zu jedem einzelnen Punkt dieser Richtlinie finden Sie auf obiger Web-Seite. Sie sollten in der Übung versuchen, die nach Ihrer Meinung zehn wichtigsten dieser Regeln zu beachten.

Beim Kompilieren sollten Sie eine möglichst hohe Warn- und Optimierungsstufe einschalten, unter Linux also bspw. mittels `g++ -Wall -Wextra -O3` kompilieren. Nehmen Sie alle Warnungen ernst und beseitigen Sie die Ursachen für diese Warnungen.

Alle Programme müssen ausreichend getestet werden. Die dafür benötigten Test-Treiber müssen zusammen mit dem Programm erstellt werden. Prüfen Sie, ob die Testfälle ausreichen. Testtreiber müssen vollständig automatisiert ablaufen und die Ergebnisse der Tests selbständig überprüfen.

2.1 Philosophy rules

- Express ideas directly in code, express intent
- Ideally, a program should be statically type safe
- Prefer compile-time checking to run-time checking
- What cannot be checked at compile time should be checkable at run time
- Catch run-time errors early
- Don't leak any resources
- Don't waste time or space
- Prefer immutable data to mutable data
- Encapsulate messy constructs, rather than spreading through the code

2.2 Function rules

Function definition rules:

- "Package" meaningful operations as carefully named functions
- A function should perform a single logical operation
- Keep functions short and simple
- If a function is very small and time-critical, declare it `inline`
- If your function may not throw, declare it `noexcept`
- For general use, take `T*` or `T&` arguments rather than smart pointers
- Prefer pure functions (pure functions always produce the same results when given the same arguments, never have side effects, never alter state)
- Unused parameters should be unnamed

Parameter passing expression rules:

- "in" parameters: pass cheaply-copied types by value and others by reference to const
- "in-out" parameters: pass by reference to non-const
- "out" output values: prefer return values to output parameters
- To return multiple "out" values, prefer returning a struct or tuple
- Prefer `T*` over `T&` when "no argument" is a valid option
- Where there is a choice, prefer default arguments over overloading

Value return semantic rules:

- Return a `T*` to indicate a position (only)
- Never (directly or indirectly) return a pointer or a reference to a local object
- Return a `T&` when copy is undesirable and "returning no object" isn't needed
- `int` is the return type for `main()`
- Return `T&` from assignment operators

2.3 Expressions and Statements

Declaration rules:

- Keep scopes small
- Declare names in for-statement initializers and conditions to limit scope
- Keep common and local names short, and keep uncommon and non-local names longer
- Avoid similar-looking names
- Avoid `ALL_CAPS` names except for macro names
- Do not reuse names in nested scopes
- Always initialize an object
- Don't introduce a variable (or constant) before you need to use it
- Don't declare a variable until you have a value to initialize it with
- Declare an object `const` or `constexpr` unless you want to modify its value later on
- Don't use a variable for two unrelated purposes
- Don't use macros for program text manipulation
- Don't use macros for constants or "functions"
- If you must use macros, give them unique names

Expression rules:

- Avoid complicated expressions
- If in doubt about operator precedence, parenthesize
- Keep use of pointers simple and straightforward
- Avoid expressions with undefined order of evaluation
- Don't depend on order of evaluation of function arguments
- Avoid "magic constants" use symbolic constants

- Avoid narrowing conversions
- Use `nullptr` rather than `0` or `NULL`
- Avoid casts
- If you must use a cast, use a named cast
- Don't cast away `const`
- Avoid `new` and `delete` outside resource management functions
- Delete arrays using `delete[]` and non-arrays using `delete`
- Don't compare pointers into different arrays
- Use the `T{e}` notation for construction
- Don't dereference an invalid pointer

Statement rules:

- Prefer a `switch`-statement to an `if`-statement when there is a choice
- Prefer a `for`-statement to a `while`-statement when there is an obvious loop variable
- Prefer a `while`-statement to a `for`-statement when there is no obvious loop variable
- Prefer to declare a loop variable in the initializer part of a `for`-statement
- Avoid `do`-statements
- Avoid `goto`
- Minimize the use of `break` and `continue` in loops
- Don't rely on implicit fallthrough in switch statements
- Use `default` to handle common cases (only)
- Make empty statements visible
- Avoid modifying loop control variables inside the body of raw `for`-loops
- Don't add redundant `==` or `!=` to conditions

Arithmetic rules:

- Don't mix signed and unsigned arithmetic
- Use unsigned types for bit manipulation
- Use signed types for arithmetic
- Don't overflow
- Don't underflow
- Don't divide by zero
- Don't try to avoid negative values by using `unsigned`

3 Bibliotheken

Viele Softwareprojekte in der heutigen Zeit verwenden Software, die von Dritten programmiert wurde. Um die Einbindung fremder Software zu üben, werden die Übungen auch externe Frameworks nutzen: `ncurses` unter Linux bzw. `pdcurses` unter Microsoft Windows für eine einfache ASCII-Grafik auf der Console, `SFML` zur grafischen Darstellung und Apache Thrift zur Kommunikation zwischen Rechnern.

3.1 ncurses

Um einfache Spiele zu implementieren, muss nicht unbedingt ein Graphical User Interface genutzt werden, die meistens recht aufwendig zu erlernen sind. Oft reicht schon ein Text User Interface wie `ncurses` unter Linux oder dessen Portierung `pdcurses` unter Microsoft Windows völlig aus.

Installation:

- Microsoft Windows: Zunächst muss das Paket `pdcurses` von einer Internet-Seite wie `https://sourceforge.net/projects/pdcurses/files/` herunter geladen und in einem Verzeichnis entpackt werden.

  Über die Systemsteuerung wird die Umgebungsvariable `PATH` angepasst, damit die Datei `pdcurses.dll` später bei der Programmausführung oder vom Debugger gefunden wird.

  Außerdem müssen das Include- und das Bibliotheksverzeichnis in den Projekteinstellungen unter Visual Studio angepasst werden. Also: Nachdem ein neues Projekt angelegt wurde, die Projekteigenschaften öffnen, z.B. mittels Rechtsklick auf den Projektnamen, und anschließend den Punkt Eigenschaften wählen. Unter Konfigurationseigenschaften und VC++-Verzeichnisse das Include- und das Bibliotheksverzeichnis angeben.

  Zum Schluss muss noch die Abhängigkeit angegeben werden, die der Linker auflösen muss. Dazu wird die lib-Datei `pdcurses.lib` in den zusätzlichen Abhängigkeiten eingetragen: Bei `Konfigurationseigenschaften` → `Linker` → `Eingabe` den Punkt `Zusätzliche Abhängigkeiten` auswählen und die Datei `pdcurses.lib` eintragen.

- Linux: Unter Ubuntu oder Debian kann mittels `apt install libncurses5-dev` das Paket `ncurses` installiert werden, bei anderen Linux-Derivaten mit den entsprechenden Installations-Tools.

Auf der Web-Seite zur Vorlesung finden Sie drei kleine Beispielprogramme: 15-Puzzle, Solitaire und Snake. Schauen Sie sich diese Beispiele an und entwickeln Sie andere Spiele wie Tic-Tac-Toe, Vier-Gewinnt, Schiffe versenken, Pacman, Bolderdash, Tetris, Autorennen, Minesweeper oder was Sie wollen. Dokumentationen zu der `curses`-Bibliothek finden Sie

bspw. unter `https://de.wikibooks.org/wiki/Ncurses`. Im wesentlichen benötigen Sie nur wenige Befehle zur Steuerung der Ausgabe:

```
#include <curses.h>
.....

int main(void) {
    int ch;                 // to store the input character

    initscr();              // initialize standard screen
    noecho();               // do not echo characters
    curs_set(0);            // make the cursor invisible
    cbreak();               // parse input per char, not at endlines
    keypad(stdscr, TRUE);   // turn on function and arrow keys
    timeout(200);           // wait only 200 milliseconds for input
    start_color();          // enable colors

    // color table: identifier, foreground color, background color
    init_pair(1, COLOR_WHITE, COLOR_BLACK);
    init_pair(2, COLOR_GREEN, COLOR_BLACK);
    init_pair(3, COLOR_RED, COLOR_MAGENTA);

    .....
    erase();                // clear screen

    ch = getch();           // read input from keyboard
    if (ch == KEY_UP) {
        .....       // handle keyboard event KEY_UP
    } else if (ch == KEY_LEFT) {
        .....       // handle keyboard event KEY_LEFT
    } else if (ch == 'q') {
        .....       // user pressed 'q' to quit the program
    } else .....    // do default action

    // move cursor to row 5 and column 10 and print string
    mvprintw(5, 10, "Points: %d", points);

    // move cursor to row 10 and column 20 and add character
    mvaddch(10, 20, ACS_DIAMOND);

    attron(A_BOLD);         // turn on attribute bold font
    attron(COLOR_PAIR(1)); // switch to color pair 1
    mvaddch(8, 20, 'O');    // move to row 8, column 20 and add 'O'
    attroff(COLOR_PAIR(1));// switch to original color pair
    attroff(A_BOLD);        // turn off attribute bold font
```

```
    refresh();              // update screen
    .....

    endwin();               // switch terminal to original output
    return 0;
}
```

Weitere Funktionen der `curses`-Bibliothek entnehmen Sie bitte den online verfügbaren Tutorials. Viel Spaß beim Programmieren von Spielen!

Verwenden unter Linux

Wenn der Quelltext in einer Datei namens `spiel.cpp` abgespeichert ist, erfolgt das Übersetzen des Quelltextes mittels `g++ spiel.cpp -o spiel -lncurses`, anschließend kann das Programm mittels `./spiel` auf der Konsole ausgeführt werden. Wir empfehlen eine Übersetzung mittels

```
g++ -Wall -Wextra spiel.cpp -o spiel -lncurses -O3
```

um wichtige Warnungen zu erhalten. Alle Warnungen sollten ernst genommen werden und der Code entsprechend umgeschrieben werden.

3.2 SFML

Die Simple and Fast Multimedia Library[1] ist ein plattformunabhängiges, objektorientiertes Open-Source Multimedia-Framework, das unter der zlib/libpng-Lizenz steht. Es ist in C++ geschrieben und greift intern auf betriebssystemspezifische Funktionen sowie externe Bibliotheken zurück. Neben C++ bietet es Anbindungen für die Programmiersprachen C, .NET, Python und anderen. Einige sehr brauchbare Tutorials finden Sie im Internet auf der Seite https://www.sfml-dev.org/tutorials/2.5/.

Im folgenden Programm wird der Text von links oben nach rechts unten verschoben. Auf die Geschwindigkeit kann mittels der Tastatur, `f` für faster und `s` für slower, Einfluss genommen werden. Mittels Mouse-Klick kann das Icon auf dem Bildschirm platziert werden. Die angegebenen Pfade bei den Methoden `loadFromFile` beziehen sich auf das aktuelle Ausführungsverzeichnis. In Zeile 14 wird das Icon `image.png` aus dem aktuellen Verzeichnis geladen, in Zeile 21 wird der absolute Pfad angegeben, der im Beispiel nicht vollständig abgedruckt ist und durch den Pfad ersetzt werden muss, wo sich die Schriftarten auf Ihrem Rechner befinden.

```
1  #include <SFML/Graphics.hpp>
2
3  int main() {
4      int speed = 5;
```

[1] https://de.wikipedia.org/wiki/Simple_and_Fast_Multimedia_Library

```
5       int row = 10, col = 50;
6
7       // Create the main window
8       sf::VideoMode mode(800, 600);
9       sf::RenderWindow window(mode, "SFML window");
10      window.setFramerateLimit(speed);
11
12      // Load a sprite to display
13      sf::Texture texture;
14      if (!texture.loadFromFile("image.png"))
15          return EXIT_FAILURE;
16
17      sf::Sprite sprite(texture);
18
19      // Create a graphical text to display
20      sf::Font font;
21      if (!font.loadFromFile("/usr/share/fonts/.../arial.ttf"))
22          return EXIT_FAILURE;
23
24      sf::Text text("Hello SFML", font, 50);
25      text.setPosition(sf::Vector2f(row, col));
26
27      // Start the game loop
28      while (window.isOpen()) {
29          // Process events
30          sf::Event event;
31          while (window.pollEvent(event)) {
32              if (event.type == sf::Event::Closed)
33                  window.close();
34              // handle mouse events
35              if (event.type == sf::Event::MouseButtonPressed) {
36                  int x = event.mouseButton.x;
37                  int y = event.mouseButton.y;
38                  sprite.setPosition(sf::Vector2f(x, y));
39              }
40              // handle keyboard events
41              if (event.type == sf::Event::KeyPressed) {
42                  if (event.key.code == sf::Keyboard::S)
43                      speed -= 1;
44                  if (event.key.code == sf::Keyboard::F)
45                      speed += 1;
46                  window.setFramerateLimit(speed);
47              }
48          }
49          text.setPosition(sf::Vector2f(++row, ++col));
```

```
50
51          window.clear();
52          window.draw(sprite);
53          window.draw(text);
54          window.display();
55      }
56
57      return EXIT_SUCCESS;
58  }
```

Verwenden unter Linux

Wenn das obige Programm in der Datei `main.cpp` gespeichert wurde, kann das Programm wie folgt übersetzt werden:

```
g++ -Wall -Wextra main.cpp -lsfml-graphics -lsfml-window -lsfml-system
```

3.3 Apache Thrift

Sowohl unter Linux als auch unter Windows muss Apache Thrift [2] kompiliert und dann installiert werden, es gibt keine fertigen Pakete zum Download. Unter Linux gehen Sie wie folgt vor.

- Aktualisieren Sie zunächst Ihr System: `sudo apt update`
- Installieren Sie dann alle für Thrift notwendigen Pakete:
  - `sudo apt install wget g++ libboost-dev libboost-test-dev automake`
  - `sudo apt install autoconf libtool libssl-dev libevent1-dev`
- Wechseln Sie in das Verzeichnis, wo Sie Thrift installieren wollen, z.B. `cd /opt`, anschließend muss das Verzeichnis für Thrift erstellt werden:
  - `sudo mkdir thrift-0.13.0`.
  - `sudo chown <your username> thrift-0.13.0`.
  - `chgrp <your group> thrift-0.13.0`.
- Laden Sie dann die aktuelle Version herunter:

  `wget https://ftp.fau.de/apache/thrift/0.13.0/thrift-0.13.0.tar.gz`
- Auspacken der Dateien: `tar xfvz thrift-0.13.0.tar.gz`
- Wechseln Sie nun in das Verzeichnis: `cd thrift-0.13.0`

[2] `https://thrift.apache.org/download`

- Hier führen Sie nun `./configure --without-test` aus. Falls dabei Fehlermeldungen erzeugt werden, müssen Sie ggf. weitere Pakete installieren.
- Zum Schluss rufen Sie `make` und `sudo make install` auf.
- Nun noch einen (symbolischen) Link auf die Library setzen, damit die Bibliothek auch vom Compiler gefunden wird:

  ```
  sudo ln /usr/local/lib/libthrift-0.13.0.so /usr/lib/libthrift-0.13.0.so
  ```