Aufgabe 1

Bearbeitungszeit: zwei Wochen (bis Montag, 20. April 2020)

Mathematischer Hintergrund: Nullstellenbestimmung eines quadratischen Polynoms

Elemente von C++: Einbinden von Header–Dateien, Variablendeklaration, arithmetische Operationen, Zuweisungen, if–Abfragen, etc.

Elemente von Linux: Arbeiten mit Editor und Compiler

Aufgabenstellung

Schreiben Sie einen Algorithmus, der *alle* Nullstellen eines quadratischen Polynoms berechnet. Dabei ist zu beachten, dass das Polynom auch linear oder konstant sein kann. Vermeiden Sie insbesondere Auslöschung und Überlauf bei den Zwischenergebnissen.

Zeichnen Sie für Ihnen Algorithmus einen Programmablaufplan mit Symbolen nach DIN 66001. Beginnen Sie erst dann damit, den Algorithmus zu implementieren.

Auslöschung und Überlauf

Falls bei einem gegebenen quadratischen Polynom $ax^2 + bx + c$ die Konstante $a \neq 0$ ist, kann das Polynom mittels Division durch a zunächst normiert werden: $x^2 + px + q$. Wendet man hierauf die „pq–Formel“ an, so erhält man als Nullstellen

$$x_{1/2} = -\frac{p}{2} \pm \sqrt{\left(\frac{p}{2}\right)^2 - q}.$$

Ist $p > 0$, so werden bei der Berechnung von x_1 zwei Größen ($-\frac{p}{2}$ und die Wurzel) mit unterschiedlichen Vorzeichen addiert. Dies kann durch Auslöschung zu ungenauen Ergebnissen führen. Dieser Effekt wird umso stärker, je kleiner $|q|$ ist, da dann beide Summanden annähernd betragsgleich sind. Bei x_2 haben beide Größen dasselbe Vorzeichen, so dass keine Auslöschung und damit kein Genauigkeitsverlust auftritt. Aus der Beziehung $q = x_1 x_2$ (Satz des Vieta) lässt sich x_1 dann ebenfalls ohne Genauigkeitsverlust bestimmen. Für $p < 0$ sind die Rollen von x_1 und x_2 zu vertauschen.

Ein weiteres Problem wird durch $\left(\frac{p}{2}\right)^2$ verursacht. Falls $|p|$ sehr groß ist, kann dieser Ausdruck bereits die größte darstellbare Zahl auf dem Rechner überschreiten, obwohl die Wurzel wieder klein genug wäre. Hier schafft ein Umschreiben der Wurzel

$$\sqrt{\left(\frac{p}{2}\right)^2 - q} \quad = \quad |p| \sqrt{\frac{1}{4} - \frac{q/p}{p}}.$$

Abhilfe. Falls $|p|$ jedoch klein ist, sollte man die linke Seite vorziehen.

Header–Dateien

Die Koeffizienten `a`, `b` und `c` des Polynoms werden durch die externe Funktion

```
void getExample(int example_id, double &a, double &b, double &c);
```

übergeben. Um in einem Modul (hier das von Ihnen zu schreibende Programm `meina1.cpp`) Funktionen eines anderen Moduls (hier die bereits kompilierte Praktikumsumgebung `unit.o`) benutzen zu können, müssen die Funktionsköpfe (Header) des zweiten Moduls in das erste eingebunden werden. Dazu stellen wir Ihnen eine Header–Datei (hier `unit.h`) zur Verfügung, in der alle Funktionsköpfe der Funktionen aufgenommen sind, die nach außen zugänglich gemacht werden sollen. Sie wird dann mittels

```
#include "unit.h"
```

in das Modul (hier `meina1.cpp`) eingebunden. Header–Dateien, die bereits das System zur Verfügung stellt, werden etwas anders eingebunden, beispielsweise bindet

```
#include <cmath>
```

den Header `cmath` ein. Genaueres findet man in der Literatur unter den Stichworten *Header-Files* und `#include`. Als Beispiel für die Struktur einer solchen Header-Datei sollten Sie sich die Datei `unit.h` anschauen.

Um Ihnen die Arbeit bei der ersten Aufgabe ein wenig zu erleichtern, geben wir Ihnen ein Programmgerüst in der Datei `meina1.cpp` vor, das Sie bitte vervollständigen. Zum Kompilieren und Linken des fertigen Programms verwenden Sie den Befehl[1]

```
g++ -g -std=c++17 -o prog1 meina1.cpp unit.o
```

Das ausführbare Programm hat dann den Namen `prog1`.

Aufbau des Programms

Zu Beginn des Programms muss die Funktion `getExample` (siehe oben) aufgerufen werden, die die Koeffizienten `a`, `b` und `c` des Polynoms $ax^2 + bx + c$ in Abhängigkeit des Wertes der Variablen `example_id` zurückgibt. Es stehen insgesamt `num_examples` Polynome zur Auswahl. Das Programm soll von allen die Nullstellen berechnen können. Die Konstante `DBL_MAX` liefert die größte darstellbare `double`–Zahl. Mit ihrer Hilfe können Sie abschätzen, ob Sie die normale pq–Formel benutzen können oder $|p|$ aus der Wurzel ausklammern müssen. Nach erfolgter Nullstellenbestimmung soll die Funktion

```
void checkSolution(int num, bool is_complex = false, double x1 = 0, double x2 = 0);
```

aufgerufen werden, um das Resultat zu beurteilen, wobei `num` die Anzahl der Nullstellen darstellt. Die Variable `is_complex` gibt an, ob die Nullstellen komplex sind oder nicht, wobei logische Werte in C++ durch `true` (wahr) und `false` (falsch) dargestellt werden. Je nach Anzahl der Nullstellen sind mehrere Fälle zu unterscheiden: Falls das Polynom identisch Null ist, so ist die Zahl der Nullstellen `infinity` (spezielle Integer–Konstante, die durch `unit.h` zur Verfügung gestellt wird). Sie können die Funktion `checkSolution` dann in der Form

```
checkSolution(infinity);
```

aufrufen. Bei keiner oder einer Nullstelle ist der korrekte Aufruf

`checkSolution(0);` bzw. `checkSolution(1, false,` *Nullstelle*`);`

[1] Eine ausführliche Liste aller Optionen des GNU C++-Compilers finden Sie mit dem Komandozeilenbefehl `info g++` und die vollständige Dokumentation unter `http://gcc.gnu.org/`.

Bei zwei reellen Nullstellen übergeben Sie bitte beide Nullstellen mit

`checkSolution(2, false,` *Nullstelle1*`,` *Nullstelle2*`);`

wobei jedoch die betragsgrößere zuerst angegeben wird. Bei zwei komplexen Nullstellen übergeben Sie bitte mit

`checkSolution(2, true,` *Realteil*`,` *Imaginärteil*`);`

den Real- und den (positiven) Imaginärteil der Nullstellen.

Programmablaufplan

In einem Programmablaufplan (PAP) oder Flußdiagramm wird ein Algorithmus grafisch dargestellt. Die DIN-Norm 66001 legt die Symbole für Programmablaufpläne fest. Auch in Normdokumenten der ISO werden Algorithmen oft in Form von Programmablaufplänen angegeben.

In der Programmierpraxis und auch im Mathematischen Praktikum hat es sich sehr oft bewährt, einen Programmablaufplan zu erstellen, bevor man mit der Implementierung eines Algorithmus beginnt. Dies soll in dieser Aufgabe am Beispiel des Algorithmus der Nullstellenberechnung eines quadratischen Polynoms geübt werden. Ein Beispiel eines Programmablaufplans finden Sie in Abbildung 1.

Programmabnahme

Denken Sie daran, eine sinnvolle Einrückung[2] Ihres Codes vorzunehmen, um die Lesbarkeit zu erhöhen. Desweiteren ist es sinnvoll, Bezeichner einheitlich zu benennen. Dabei ist es (fast) egal, auf welchen Konvention Sie sich einigen, solange sie einheitlich umgesetzt wird. Eine Möglichkeit ist dabei der Google C++ Style Guide[3], an dem sich die Aufgabenstellungen und Codegerüste orientieren.

Wird das Ergebnis bei allen Beispielen durch die Praktikumsumgebung akzeptiert, so lade man den Quellcode des Programms und den Programmablaufplan im RWTHmoodle-Lernraum hoch, erkläre den Betreuern über zoom[4] das Programm und lasse sich die Lösung der Aufgabe testieren.

[2] Zum automatischen Einrücken Ihres Codes können Sie auch die Programme `indent` oder `clang-format` verwenden. Eine Hilfeseite erhalten Sie durch den Aufruf von `man indent` bzw. unter `https://clang.llvm.org/docs/ClangFormat.html`.

[3] `https://google.github.io/styleguide/cppguide.html#Naming`

[4] `https://zoom.us`

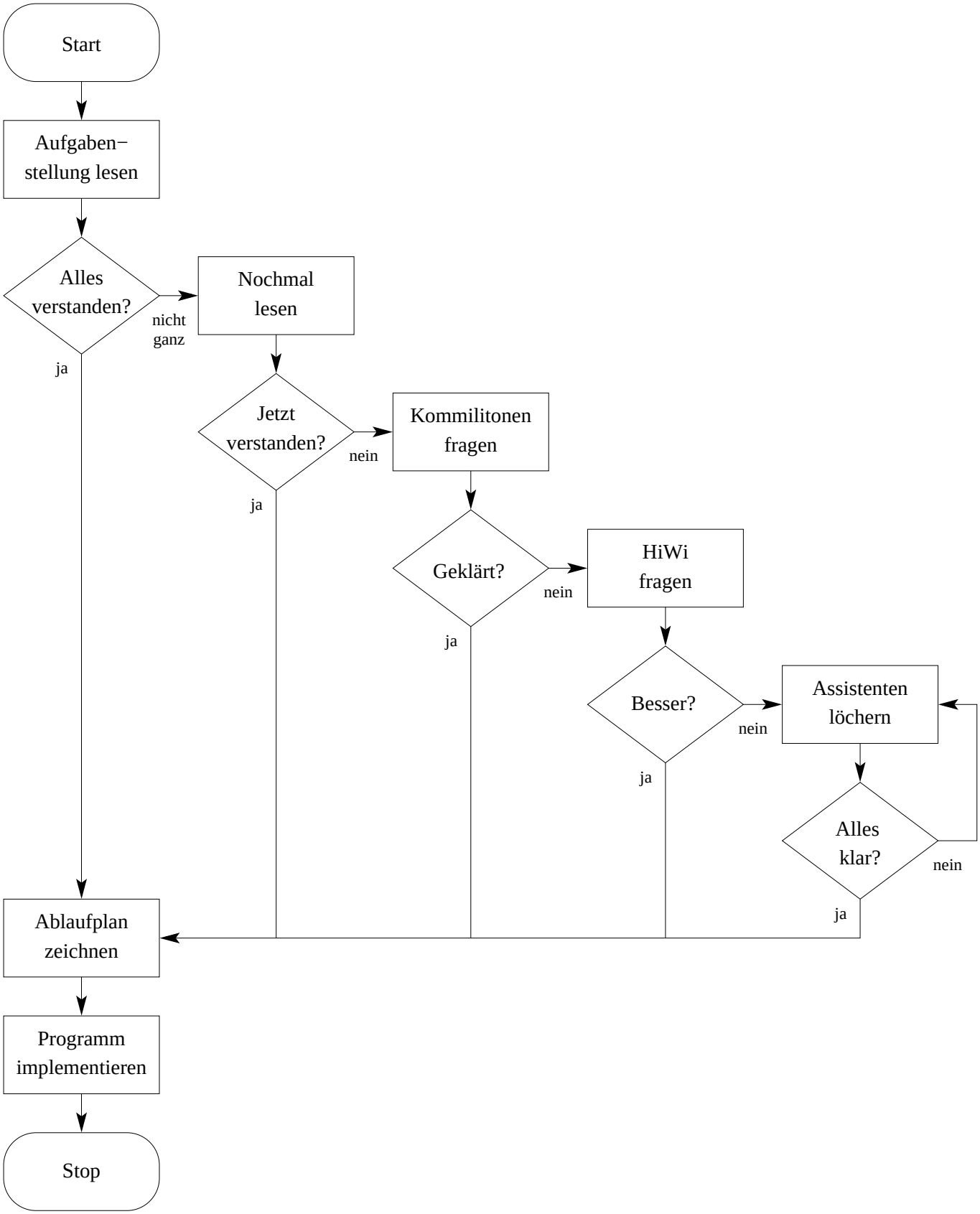

Abbildung 1: Beispiel eines Programmablaufplans: Vorgehen zur Lösung der ersten Aufgabe