Programmieren in C++

SS 2022

Vorlesung 1, Dienstag 26. April 2022

(Ein erstes Programm + das ganze Drumherum)

Prof. Dr. Hannah Bast
Algorithmen und Datenstrukturen
Institut für Informatik
Universität Freiburg

Blick über die Vorlesung heute

UNI FREIBURG

- Organisatorisches
  - Ablauf — Vorlesungen, Übungen, Projekt
  - Prüfungstechnisches — Punkte, Note, ECTS & Aufwand
  - Art der Vorlesung — Voraussetzungen, Lernziel, Stil
- Inhalt
  - Unser erstes Programm — Ist eine gegebene Zahl prim?
  - mit allem Drumherum — Kompilieren, Unit Test, Checkstyle Makefile, SVN, Jenkins

Übungsblatt 1: Kleine Aufgabe zum Collatz-Problem

Der Code ist relativ einfach und kurz, die meiste Arbeit beim ersten Mal wird das ganze Drumherum sein

Ablauf 1/5

UNI FREIBURG

- Vorlesungen
  - Jeden Dienstag von **14:00** – 15:45 Uhr
    - Insgesamt 12 Termine (zwischen 26. April und 26. Juli)
    - Sie können wählen: Präsenz, Zoom, Aufzeichnungen

      Details dazu auf der nächsten Folie und auf dem Wiki
    - Technik: Frank Dal-Ri, Schnitt: Alexander Monneret
  - Assistent der Vorlesung: Johannes Kalmbach
  - Auf unserem Wiki finden Sie alle Kursmaterialien
    - Aufzeichnungen, Folien, Übungsblätter, Code aus der Vorlesung + evtl. zusätzliche Hinweise, Musterlösungen
    - Auch im SVN, Unterordner /public (außer die Aufzeichnungen)

Ablauf 2/5

- Drei Möglichkeiten, die Vorlesung zu konsumieren
  - In Präsenz, im SR 01-019/013 in Gebäude 101

    Platz für ca. 50 Leute, das könnte am Anfang eng werden, aber spätestens nach ein paar Wochen völlig ausreichend
  - Über Zoom, Einwahldaten siehe Wiki

    Auch über Zoom haben Sie die Möglichkeiten live Fragen zu stellen, durch Wortmeldung oder über den Chat
  - Über die Vorlesungsaufzeichnungen

    Werden im Regel am selben Tag noch geschnitten, auf YouTube hochgeladen und auf dem Wiki verlinkt
  - Sie können sich jede Woche neu entscheiden, in welchem Format Sie teilnehmen wollen

Ablauf 3/5

UNI FREIBURG

- Übungsblätter
  - Die Übungen sind der wichtigste Teil der Veranstaltung
  - Sie bekommen jede Woche ein Übungsblatt, insgesamt 10
  - Das Übungsblatt können Sie bearbeiten wo und wann Sie wollen (bis zur Abgabefrist), aber Sie müssen es **selber** machen!

    Sie können gerne zusammen über die Übungsblätter nachdenken, gemeinsam diskutieren, etc. … aber die Programme müssen Sie dann zu **100%** selber schreiben

    Auch das teilweise Übernehmen gilt als Täuschungsversuch, siehe das 10. Gebot, das auf dem Wiki und dem Ü1 verlinkt ist

    Wenn Sie gegen diese klaren Vorgaben verstoßen und sich auf ein "Missverständnis" berufen, werden wir das nicht akzeptieren

Ablauf 4/5

UNI FREIBURG

- Das Projekt
  - Am Ende der Veranstaltung gibt es eine etwas größere Programmieraufgabe, genannt **das Projekt**
  - Umfang ca. 4 Übungsblätter
  - Weniger Vorgaben als bei den Übungsblättern
  - Fängt schon zwei Wochen vor Vorlesungsende an
  - Wer schnell ist, kann dann schon in der letzten Vorlesungswoche mit **das Projekt** fertig werden

Ablauf 5/5

UNI FREIBURG

- "Übungsgruppen"
  - Sie bekommen jede Woche Feedback zu Ihren Abgaben

    Von Ihrer:m Tutor:in über unser SVN, siehe Folie 26
  - Für Fragen aller Art gibt es ein **Forum**

    Siehe Link auf dem Wiki + mehr dazu auf Folie 28
  - Bei Problemen, die sich nicht über das Forum lösen lassen, können Sie Ihren Tutor um ein persönliches Treffen bitten

    Die Zuordnung zu den Tutor:innen geschieht aber erst nächsten Dienstag, nach der Abgabe des Ü1

    Deshalb diese Woche eine Auswahl von **drei** Terminen (über Zoom) für Rückfragen zum Drumherum, siehe Wiki

Prüfungstechnisches 1/3

UNI FREIBURG

- Punkte
  - Sie bekommen wunderschöne Punkte, maximal 20 pro Übungsblatt, das sind maximal 200 Punkte für Ü1 – Ü10
  - Für **das Projekt** gibt es noch mal maximal 100 Punkte
  - Macht insgesamt 300 Punkte
  - Für das Ausfüllen des Evaluationsbogens am Ende gibt es 20 Punkte, mit denen Sie die Punktezahl des schlechtesten der Übungsblätter 1 – 10 ersetzen können

    Man kann also auch einfach ein Übungsblatt ausfallen lassen, z.B. wegen Krankheit oder Kino oder WM
  - Alternativ gibt es +10 Punkte für das Projekt, wir schauen von uns aus, was günstiger für Sie ist

Prüfungstechnisches 2/3

UNI FREIBURG

- Gesamtnote
  - Die ergibt sich linear aus der Gesamtpunktzahl am Ende

    150 – 164: 4.0; 165 – 179: 3.7; 180 – 194: 3.3
    195 – 209: 3.0; 210 – 224: 2.7; 225 – 239: 2.3
    240 – 254: 2.0; 255 – 269: 1.7; 270 – 284: 1.3
    285 – 300: 1.0

  - **Außerdem:** Zum Bestehen müssen mindestens 100 Punkte in den Übungen (Ü1 – Ü10) und mindestens 50 Punkte für das Projekt erreicht werden
  - **Außerdem 2:** Sie müssen sich mindestens einmal mit Ihrem Tutor / Ihrer Tutorin treffen, dazu mehr in einer der späteren Vorlesungen

Prüfungstechnisches 3/3

- ECTS Punkte und Aufwand
  - Seit PO 2018: **6 ECTS** Punkte  früher: 4 ECTS Punkte
  - Das sind 6 x 30 = 180 Arbeitsstunden insgesamt
  - Davon 2/3 für die ersten 10 Vorlesungen + Übungen

    Also ca. 8 - 12 Stunden Arbeit / Woche (inkl. Vorlesung)
  - Und etwa 1/3 für das Projekt und die dazugehörigen (letzten beiden) Vorlesungen + Übungen

    Wer schon Vorkenntnisse hat, wird weniger Arbeit haben

    Wer in "Info I" oder "Einführung in die Informatik" keine Übungen gemacht hat, wird mehr Arbeit haben

Art der Vorlesung 1/6

UNI FREIBURG

- Voraussetzungen
  - Sie wissen schon, wie man "im Prinzip" programmiert, z.B.:
    - die Zahlen von 1 bis 10 ausgeben
    - berechnen, ob eine gegebene Zahl prim ist
    - Einfache KI mit Ich-Bewusstsein 
  - Verständnis einiger Grundkonzepte
    - Variablen, Funktionen, Schleifen, Ein- und Ausgabe
- Wenn Ihnen das alles gar nichts sagt
  - ... können Sie trotzdem mitmachen, es wird aber dann mehr Arbeit für Sie, als es den ECTS Punkten entspricht

Art der Vorlesung 2/6

UNI FREIBURG

- Lernziel
  - Programmieren in C++ nach den Regeln der Kunst
    … im Umfang von bis zu 1000 Zeilen
  - Einfaches objektorientiertes Programmieren
  - Gute Struktur, gute Namen, gute Dokumentation
  - Verwendung von Standardtools und –techniken:
    - Unit Tests gtest
    - Stylechecker cpplint.py
    - Build System make und Jenkins
    - Repository SVN

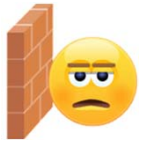
Siehe Post auf dem Forum: "Warum wir SVN statt Git verwenden"

Art der Vorlesung 3/6

UNI FREIBURG

- Die Vorlesung ist eher **breit** als **tief**
  - Die Aufgaben sind eher einfach
  - Die Konstrukte / Konzepte die wir verwenden auch
  - Sie sollen vor allem lernen, **guten Code** zu schreiben
  - Mit allem Drum und Dran
  - Das ist nämlich genau das, was viele Leute, die es sich selber beibringen oder beigebracht haben, nicht können
  - Außerdem macht Programmieren viel mehr Spaß, wenn man es richtig lernt und auch das Drumherum beherrscht

    Sonst hackt man die Sachen immer irgendwie zusammen und ist vor allem am Debuggen, statt am Programmieren



Art der Vorlesung 4/6

UNI FREIBURG

- Art der Vermittlung
  - Ich werde das meiste exemplarisch vormachen
  - Für die Details gibt es genügend Referenzmanuale, siehe jeweils die letzte Folie jeder Vorlesung
  - Und Sie kennen ja **Google und Co**
  - Ich werde vor allem immer das erklären, was Sie auch gerade brauchen (für das jeweilige Übungsblatt)
- Fragen, Fragen, Fragen
  - Selber ausprobieren, aber nicht zu lange, dann fragen!

    Hier in der Vorlesung oder über das Forum → Folie 28

Art der Vorlesung 5/6

UNI FREIBURG

- Wir machen hier alles bewusst sehr "low-level"
  - Linux, Kommandozeile, Texteditor, Makefile
  - Was das konkret heißt, sehen Sie gleich
  - So lernt man am besten was tatsächlich passiert
  - Aufwändige Entwicklungsumgebungen / IDEs (z.B. Eclipse) sind was für später, wenn man den "low level" verstanden hat

    Wer sich auskennt, kann eine IDE benutzen, die Sachen müssen aber **trotzdem** "low-level" in unser SVN hochgeladen werden (insbesondere mit Makefile)

    Das ist dann aber mehr Arbeit anstatt weniger, eine IDE lohnt den Aufwand erst für größere Projekte

Art der Vorlesung 6/6

UNI FREIBURG

- Linux
  - Sie können Ihren eigenen Linux-Rechner verwenden oder auf einem der Pool-Rechner arbeiten
  - Für Teilnehmer aus anderen Universen steht auf dem Wiki außerdem ein **Linux-Image** zur Verfügung

    Da sind g++, SVN, gtest und alles, was man für diese Vorlesung so braucht, schon vorinstalliert
  - Unter Windows kann man auch sehr gut mit dem WSL (Windows Subsystem Linux) arbeiten

    Das verhält sich für unsere Zwecke, bis auf ein paar Randfälle, genau so wie ein Linux System
  - Für MacOS bieten wir keine Unterstützung 

Unser erstes Programm 1/12

UNI FREIBURG

- Wir schauen uns ein einfaches Problem an
  - Ich werde Ihnen ein Problem vorstellen und wir werden das dann **live** in ein Programm umsetzen
  - Es kommt dabei weniger auf das Problem an, sondern vor allem um das ganze **Drumherum**
  - Passen Sie gut auf, Sie werden viel davon gut für das Übungsblatt 1 gebrauchen können

    Insbesondere werden Sie viele der Fehler, die uns jetzt gleich begegnen, beim Selbermachen wiedertreffen

Unser erstes Programm 2/12

UNI FREIBURG

- Ist eine gegebene natürliche Zahl n prim
  - Sehr einfacher Algorithmus:

    Wir probieren einfach für alle natürlichen Zahlen i mit $2 \leq i \leq \sqrt{n}$ aus, ob sie ein Teiler von n sind

  - Warum reicht es aus, nur $i \leq \sqrt{n}$ zu testen, anstatt alle $i \leq n$?

    Wenn es einen Teiler i mit $\sqrt{n} < i < n$ gibt, dann ist auch n / i ein Teiler und $n / i < \sqrt{n}$ $\quad i > \sqrt{n} \Rightarrow \frac{n}{i} < \frac{n}{\sqrt{n}} = \sqrt{n}$

    Das ist ein gutes Beispiel dafür, wie einem (oft einfache) Mathematik dabei hilft, effizienteren Code zu schreiben

z.B. $n = 42$, $i = 21$ ist ein Teiler $\Rightarrow 42/21 = 2$ ein Teiler

$\sqrt{42} \approx 6..$

Unser erstes Programm 3/12

UNI FREIBURG

- Unser Programm, Version 1
  - Wir schreiben erstmal unseren gesamten Programmcode in eine Textdatei

    Prime.cpp
  - Sie können dafür einen beliebigen Texteditor verwenden
  - Ich benutze in der Vorlesung **vim**

    Wer auch vim benutzen möchte, findet auf dem Wiki eine Konfigurationsdatei (.vimrc) dafür, mit vielen Shortcuts, die ich auch in der Vorlesung öfter benutze

Unser erstes Programm 4/12

UNI FREIBURG

- Kompilieren
  - Wir benutzen den Gnu C++ Compiler, kurz **g++**

    g++ –o Prime Prime.cpp

    Ich benutze in der Vorlesung die g++ **Version 9.4.0**

    Auf dem Linux Image (siehe Wiki) ist es dieselbe Version

    Andere oder experimentelle Versionen auf eigene Gefahr!
  - Der Befehl oben erzeugt Maschinencode, den man dann wie folgt ausführen kann

    ./Prime

    Ohne die –o Option würde das ausführbare Programm einfach "a.out" heißen, was jetzt nicht so eingängig ist

Unser erstes Programm 5/12

UNI FREIBURG

- Unser Programm, Version 2
  - Wir schreiben jetzt zwei Dateien (warum sehen wir gleich)

    Prime.cpp
    PrimeMain.cpp
  - Die erste Datei enthält nur unsere Funktion
  - Die zweite Datei enthält das restliche Programm und liest zu Beginn die erste Datei ein

    ```
    #include "./Prime.cpp"
    ```

In Vorlesung 2 werden wir sehen, dass man das nicht so machen sollte und warum, aber für heute (und für das Ü1) ist das völlig in Ordnung

Unser erstes Programm 6/12

- Unit Tests
  - Wir schreiben jetzt noch eine dritte Datei

    PrimeTest.cpp

  - Diese enthält eine Funktion, die unsere Funktion testet und ein generisches main, das einfach alle Tests ausführt

```
#include <gtest/gtest.h>
#include "./Prime.cpp"

TEST(PrimeTest, checkIfPrime) { … }

int main() {
  ::testing::InitGoogleTest();
  return RUN_ALL_TESTS();
}
```

Unser erstes Programm 7/12

UNI FREIBURG

- Überprüfung des Stils
  - Bisher haben wir "nur" versucht, unser Programm zum Laufen zu bringen
  - Selbstverständlich sollte unser Code auch schön und für anderen Menschen gut lesbar sein
  - Es gibt ein extra Programm dafür, wir benutzen **cpplint.py**
  - Um damit den Stil aller .cpp Dateien zu überprüfen, können wir einfach schreiben

    ```
    python3 cpplint.py *.cpp
    ```

  - Man bekommt dann sehr detaillierte und in aller Regel selbsterklärende Fehlermeldungen zum Stil des Codes

Unser erstes Programm 8/12

UNI FREIBURG

- Makefile und make
  - Woher wissen andere, wie Sie unseren Code kompilieren, testen, oder seinen Stil überprüfen?
  - Wir schreiben eine Datei **Makefile** mit folgender Syntax

```
<target>:
        <Befehl 1>
        <Befehl 2>
        ...
```

Achtung: jede der Befehlszeilen muss mit einem TAB anfangen!

  - Wenn man dann in dem Verzeichnis, in dem diese Datei steht **make <target>** ausführt, werden einfach die entsprechenden Befehle ausgeführt

    make kann noch viel mehr und das wird noch nützlich für uns sein… mehr dazu in den nächsten Vorlesungen

Unser erstes Programm 9/12

UNI FREIBURG

- Daphne
  - Daphne ist unser Kursverwaltungssystem, das mir, den Tutoren, und hoffentlich auch Ihnen das Leben leichter macht
  - **Registrieren Sie sich bitte** nach der Vorlesung dort
  - Sie kriegen dann automatisch Zugang zu

    SVN, Jenkins, Forum … siehe nächste drei Folien

    Diese Subsysteme sind per se unabhängig von Daphne und in Daphne "nur" übersichtlich zusammengefasst
  - Es läuft alles über einen Username + Passwort, nämlich das von Ihrem Uni-Account (Initialen + Zahl)

    Das läuft über den LDAP-Server vom Rechenzentrum, wir erfahren also Ihr Passwort nicht

Unser erstes Programm 10/12

UNI FREIBURG

- SVN (Subversion)
  - Ein SVN Repository ist einfach ein Verzeichnisbaum mit Dateien, die bei uns zentral auf einem Rechner liegen
  - Jeder, der sich (via Daphne) bei uns registriert, hat ein Unterverzeichnis dort → URL siehe Ihre Daphne-Seite
  - Sie bekommen eine Kopie dieses Verzeichnisses mit

    svn checkout <URL> --username=<Ihr RZ Username>
  - In Ihrer Arbeitskopie können Sie dann Sachen ändern, Unterordner und Dateien hinzufügen, etc.
    - svn add <file name> fügt eine Datei erstmals hinzu
    - svn commit <file name> lädt die Änderungen zu uns hoch

      (svn commit ohne Argument lädt alle Änderungen hoch)

Unser erstes Programm 11/12

UNI FREIBURG

- Jenkins
  - Mit Jenkins können Sie überprüfen, ob die Version Ihres Codes, die Sie zu uns hochgeladen haben, auch funktioniert
  - Jenkins schaut dazu einfach nach dem Makefile und führt dann die folgenden Befehle aus

make clean	Löscht Nebenprodukte
make compile	Kompiliert ihren Code
make test	Führt die Unit Tests aus
make checkstyle	Prüft den Stil Ihres Codes

  - Sie bekommen dann angezeigt, ob es funktioniert hat, und auf Wunsch auch die komplette Ausgabe

Unser erstes Programm 12/12

UNI FREIBURG

- Forum
  - Machen Sie bitte regen Gebrauch davon und haben Sie **keine Hemmungen**, Fragen zu stellen

    Insbesondere wenn Sie bei einem Fehler mit eigenem Nachdenken und Google und Co nicht weiterkommen

    Gerade in C++ kann man mit Kleinigkeiten Stunden verbringen, was dann sehr frustrierend ist
  - Geben Sie sich aber gleichzeitig Mühe, Ihre Fragen möglichst konkret und genau zu stellen

    Eine kurze Anleitung dazu finden Sie in den 10 Geboten und eine ausführlichere auf einer eigenen Wikiseite

Hinweise für das Ü1 1/4

UNI FREIBURG

- Die Collatz-Funktion — Lothar Collatz, 1937
  - Die sogenannte Collatz-Funktion C : **N** → **N** sei definiert als

    Falls n ungerade : C(n) = 3n + 1
    Falls n gerade : C(n) = n/2
  - Frage: Wie oft muss man die Funktion C auf ein gegebenes n ∈ **N** anwenden, bis man bei der Zahl 1 landet?

    Beispiel: 10 →

    Beispiel: 11 →

Hinweise für das Ü1 2/4

UNI FREIBURG

- Das Collatz-Problem
  - Das folgende Problem ist bis heute ungelöst:

    Wird für jede natürlich Zahl irgendwann die 1 erreicht?

    Man vermutet ja, aber bisher konnte es noch niemand beweisen (und man rät auch jeder:m Mathematiker:in, sich bloß nicht damit zu beschäftigen)
  - Für das Ü1 sollen Sie einfach für eine gegebene Zahl berechnen, ob man innerhalb einer vorgegebenen Zahl von Iterationen die 1 erreicht, und wenn ja in wie vielen

Hinweise für das Ü1 3/4

UNI FREIBURG

- Alternative Aufgabe (für die Unterforderten)
  - Wem die Aufgabe zu einfach ist, kann **alternativ** folgendes Problem lösen:

    Sei $c(n)$ die Anzahl der Iterationen, die man für eine gegebene Zahl n benötigt

    Schreiben Sie ein Programm, das für ein gegebenes n $\max_{1 \le n' \le n} c(n')$ berechnet, sowie das n' für das dieses Maximum erreicht wird

    Die Aufgabe ist auch deswegen schön, weil sie ausnutzt, dass C bzw. C++ eine sehr effiziente Sprache ist

    Versuchen Sie, diesen Wert für ein möglichst großes n zu berechnen

Hinweise für das Ü1 4/4

UNI FREIBURG

- Arbeitsaufwand
  - Sie haben **eine Woche** Zeit
  - Wie gesagt: die meiste Arbeit wird das Drumherum sein
  - Fangen Sie deswegen bitte **rechtzeitig** an und fragen Sie auf dem Forum, wenn Sie nicht weiterkommen

    Oder nehmen Sie an einer der drei Fragestunden teil (über Zoom, Einwahldaten und Zeiten siehe Wiki)
  - Wenn es Probleme mit der Installation des Google Test Framework gibt, gehen Sie das bitte als Letztes an

Literatur / Links

- SVN
  - https://subversion.apache.org
  - Außerdem kurze Einführung dazu auf dem Wiki
- Google Test
  - https://github.com/google/googletest
  - Außerdem Installationsanleitung dazu auf dem Wiki
- C++
  - https://www.cplusplus.com/doc/tutorial (einfacher)
  - https://en.cppreference.com (ultimative Referenz)
  - Zu Beginn verwenden wir nur elementare Sachen und schreiben auch erstmal im Wesentlichen nur C