Programmieren in C++

SS 2018

Vorlesung 1, Dienstag 17. April 2018

(Ein erstes Programm + das ganze Drumherum)

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

UNI 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 — Approximation von Pi

    mit allem Drumherum — Kompilieren, Unit Test, Checkstyle Makefile, SVN, Jenkins

Übungsblatt 1: Approximation der Eulerschen Zahl

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

Ablauf 1/4

UNI FREIBURG

- Vorlesungen
  - Jeden Dienstag von 14.15 - 15.45 Uhr im HS 026
    - Insgesamt 12 Termine (keine VL am 1. Mai und 22. Mai)
  - Die Vorlesungen werden live gestreamed und aufgezeichnet
    - Folien + Audio + Video
    - Technik: Frank Dal-Ri, Schnitt: Alexander Monneret
  - Assistent der Vorlesung: Axel Lehmann
  - 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 2/4

UNI FREIBURG

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

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

    Auch das teilweise Übernehmen gilt als Täuschungsversuch, siehe auch die 10 Gebote auf Seite 3 vom 1. Übungsblatt

Ablauf 3/4

UNI FREIBURG

- Das Projekt
  - Am Ende der Veranstaltung gibt es eine etwas größere Programmieraufgabe
  - 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 dem Projekt fertig werden

Ablauf 4/4

UNI FREIBURG

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

    Von Ihrer*m Tutor*in über unser SVN, siehe Folie 25
  - Für Fragen aller Art gibt es ein **Forum**

    Siehe Link auf dem Wiki + mehr dazu auf Folie 27
  - Am Anfang bieten wir auch einige Fragestunden an

    Erfahrungsgemäß werden die dann nach einigen Wochen nicht mehr in Anspruch genommen
  - Bei Problemen, die sich nicht über das Forum lösen lassen, können Sie Ihren Tutor um ein persönliches Treffen bitten

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

UNI FREIBURG

- ECTS Punkte und Aufwand
  - Informatik / ESE / MST / BOK: 4 ECTS Punkte
  - Das sind 4 x 30 = 120 Arbeitsstunden insgesamt
  - Davon 80 Stunden für die ersten 10 Vorlesungen + Übungen
    - also etwa 8 Stunden Arbeit / Woche
    - also etwa 6 Stunden pro Übungsblatt
  - Bleiben 40 Stunden 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

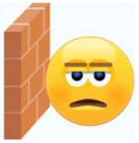
- Lernziel
  - Programmieren in C++ nach den Regeln der Kunst
    … im Umfang von 500 – 1000 Zeilen
  - Einfaches objektorientiertes Programmieren
  - Gute Struktur, gute Namen, gute Dokumentation
  - Verwendung von Standardtools und –techniken:
    - Unit Tests gtest
    - Stylechecker cpplint.py
    - Build System make und Jenkins
    - Repository SVN

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 27

Art der Vorlesung 5/6

UNI FREIBURG

- Wir machen hier alles ganz "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 oder NetBeans) 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

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
  - Wer partout unter Windows arbeiten will:

    Windows 10: hat eine integrierte Linux-Shell (Ubuntu)

    Windows 8.0: Cygwin (auf eigene Gefahr)

    Windows 3.0: **sie haben da irgendwas verpasst**

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 gebrauchen können

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

Unser erstes Programm 2/12

UNI FREIBURG

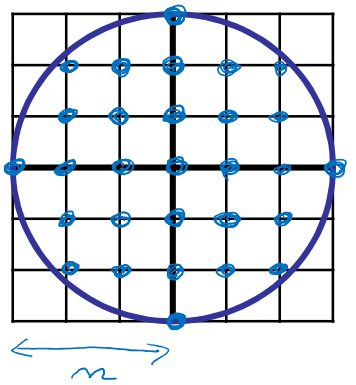
- Approximieren der Zahl **Pi** über die Kreismethode:
  - Sehr einfacher Algorithmus:

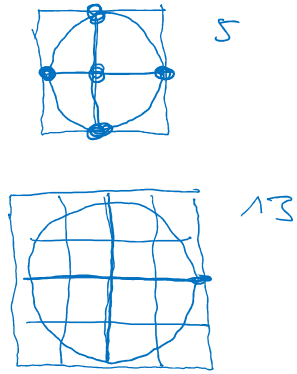
    Berechne die Anzahl aller ganzzahligen Gitterpunkte (x, y) mit $|x|, |y| \leq n$, die innerhalb des Kreises mit Radius n um den Ursprung liegen (also $x^2 + y^2 \leq n^2$)

    Für $n \to \infty$ ist der Anteil dieser Gitterpunkte dann das Verhältnis der Fläche des Kreise ($\pi \cdot n^2$) zur Fläche des Gitters ($4n^2$) ... also $\pi / 4$

n = 3

im Kreis : 29





Unser erstes Programm 3/12

UNI FREIBURG

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

    ApproximationOfPi.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 ApproximationOfPi ApproximationOfPi.cpp

    Ich benutze in der Vorlesung die g++ **Version 5.4.0** … das Linux Image (auf dem Wiki) ebenso

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

    ./ApproximationOfPi

    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)

    ApproximationOfPi.cpp
    ApproximationOfPiMain.cpp
  - Die erste Datei enthält nur unserer Funktion
  - Die zweite Datei enthält das restliche Programm und liest zu Beginn die erste Datei ein

    ```
    #include "./ApproximationOfPi.cpp"
    ```

    In Vorlesung 2 werden wir sehen, dass das nicht optimal ist, aber für heute (und das Ü1) ist das völlig in Ordnung

Unser erstes Programm 6/12

UNI FREIBURG

- Unit Tests
  - Wir schreiben jetzt noch eine dritte Datei

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

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

TEST(ApproximationOfPi, approximatePi) { ... }

int main(int argc, char** argv) {
  ::testing::InitGoogleTest(&argc, argv);
  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 extra Programm dafür, wir benutzen **cpplint.py**
  - Um damit den Stil aller .cpp Dateien zu überprüfen, können wir einfach schreiben

    ```
    python 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 es 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 vom RZ, das Sie auch für Anmeldungen etc. nutzen

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

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 findet sich auf Seite 3 des Ü1, eine etwas längere Anleitung auf dem Wiki

Hinweise für das Ü1 1/3

UNI FREIBURG

- Approximation der Eulerschen Zahl
  - Die Eulersche Zahl lässt sich sehr einfach über eine unendliche Reihe ausdrücken

    $e = \Sigma_{k=0..\infty} 1 / k! = 1 + 1/2 + 1/6 + 1/24 + \ldots$
  - Für gegebenes n, kann man einfach die ersten n Summanden dieser Summe addieren
  - Da k! für wachsende k sehr schnell sehr klein wird, konvergiert diese Summe sehr schnell

    Für das Ü1 reicht schon n = 15

Hinweise für das Ü1 2/3

UNI FREIBURG

- Approximation der Eulerschen Zahl, Variante
  - Wem diese Aufgabe algorithmisch zu einfach ist, kann auch folgendes Verfahren ausprobieren

    Sei $f(x) = \int_{1..x} 1/t \, dt = \ln x$

    Dann ist e ist gerade die Zahl x, für die $f(x) = 1$

    Ein Integral lässt sich beliebig genau durch eine Summe approximieren

    Man summiert dann einfach so lange auf, bis die Summe ≥ 1 ist und ermittelt die Approximation für e aus der Anzahl der Summanden

    Wie gesagt, nur wenn Sie Lust haben und Ihnen die Variante von der Folie vorher zu langeweilig ist

Hinweise für das Ü1 3/3

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
  - Wenn es Probleme mit der Installation des Google Test Framework gibt, gehen Sie das bitte als Letztes an
  - Fragestunden: zwei Termine diese Woche

    Donnerstag und **Montag**, jeweils **13 – 14 Uhr**

    Gebäude 51, 2. OG, Raum 028 (Büro Prof. Bast)

Literatur / Links

UNI FREIBURG

- SVN
  - http://subversion.apache.org/
  - Außerdem kurze Einführung dazu auf dem Wiki
- Google Test
  - https://github.com/google/googletest
  - Außerdem Installationsanleitung dazu auf dem Wiki
- C++
  - http://www.cplusplus.com/doc/tutorial/
  - Jetzt am Anfang brauchen wir nur elementare Sachen