Lehrstuhl für Algorith.
und Datenstrukturen
Prof. Dr. Hannah Bast
Axel Lehmann

Programmieren in C++
SS 2018

http://ad-wiki.informatik.uni-freiburg.de/teaching

UNI FREIBURG

Übungsblatt 1

Abgabe bis Dienstag, den **24. April** um **12:00 Uhr**

Aufgabe 0

Melden Sie sich bei Daphne, unserem Kursverwaltungssystem, an. Den Link dazu finden Sie auf dem Wiki zum Kurs (siehe URL in der Kopfzeile von diesem Übungsblatt). Auf Ihrer Daphne Seite finden Sie insbesondere die URL Ihres Ordners in unserem SVN Repository für diese Veranstaltung.

Aufgabe 1 (5 Punkte)

Schreiben Sie eine Funktion *approximateEulersNumber(int n)* (in einer Datei *ApproximateEulersNumber.cpp*), die die Eulersche Zahl mit Hilfe der unendlichen Reihe $\sum_{k=0}^{\infty} 1/k!$ approximiert, indem sie einfach die ersten n Summanden aufaddiert. Die Fakultät sollte nicht für jeden Summand neu berechnet werden, sondern mit Hilfe des Wertes des vorherigen Summanden. Orientieren Sie sich bezüglich der Struktur und Syntax an der entsprechenden Datei aus der Vorlesung.

Variante: wenn Sie schon mehr Programmiererfahrung haben und Ihnen diese Aufgabe zu langweilig ist, approximieren Sie stattdessen die Eulersche Zahl über eine Approximation des Integrals $\int_1^y 1/x\,dx$ durch eine Summe. Wie in der Vorlesung erklärt, hat dieses Integral für $y = e$ den Wert 1. Das Argument Ihrer Funktion sollte dann der Kehrwert der Schrittweite der Summenapproximation des Integrals sein.

Aufgabe 2 (5 Punkte)

Schreiben Sie ein lauffähiges Programm (in einer Datei *ApproximateEulersNumberTest.cpp*) mit einem Unit Test für die Funktion aus Aufgabe 1. Orientieren Sie sich bezüglich der Struktur und Syntax an dem entsprechenden Programm aus der Vorlesung.

Die Anforderungen an den Unit Test sind unter Punkt 5 auf Seite 3 dieses Übungsblattes beschrieben. Diese Anforderungen gelten auch für alle weiteren Übungsblätter sowie das Projekt.

[es geht noch weiter]

Aufgabe 3 (5 Punkte)

Schreiben Sie ein lauffähiges Programm (in einer Datei *ApproximateEulersNumberMain.cpp*), das Ihre Funktion aus Aufgabe 2 mit einem geeigneten Wert für n aufruft und eine schöne und verständliche Ausgabe produziert. Der Wert für n sollte so gewählt werden, dass das Programm höchstens wenige Sekunden läuft und dabei die Approximation so gut wie möglich ist. Orientieren Sie sich bezüglich der Struktur und Syntax an dem entsprechenden Programm aus der Vorlesung.

Aufgabe 4 (5 Punkte)

Erstellen Sie eine Datei *Makefile* nach dem Vorbild der Datei aus der Vorlesung. Laden Sie alle relevanten Dateien in einem Unterordner *blatt-01* von Ihrem Verzeichnis in unser SVN hoch. Die Datei *cpplint.py* können Sie aus dem Vorlesungscode übernehmen. Sie sollte nicht im Ordner *blatt-01* stehen, sondern in dem Ordner obendrüber (so dass Sie sie für alle Übungsblätter benutzen können).

Stellen Sie sicher, dass auf Jenkins alles ohne Fehler durchläuft. Beachten Sie dabei unbedingt die Anmerkungen auf der dritten Seite dieses Übungsblattes. Diese Anmerkungen sind wichtig und gelten auch für alle weitere Übungsblätter dieser Veranstaltung.

Lesen Sie auch die diversen Anleitungen, insbesondere zum SVN, auf dem Wiki. Verstehen Sie, dass das SVN ein Versionsmanagementsystem ist. Sie können gerne auch schon frühe Versionen von Ihrer Lösung hochladen und schauen was passiert. Das kann sogar von Vorteil sein, wenn sie in einer späteren Version etwas verschlimmbessern, was in einer vorherigen Version schon einmal funktioniert hat. Das SVN merkt sich alle Version und sie haben auch jederzeit Zugriff auf alle Versionen. Relevant für die Korrektur ist nur die letzte Version, die sie vor dem Abgabetermin (siehe oben) hochgeladen haben.

Fügen Sie dem Unterordner *blatt-01* außerdem eine Textdatei *erfahrungen.txt* hinzu und laden Sie auch diese Datei hoch. Beschreiben Sie dort in ein paar wenigen(!) Sätzen Ihre Erfahrungen mit diesem Übungsblatt und der Vorlesung dazu. Insbesondere: Wie lange haben Sie ungefähr gebraucht und an welchen Stellen gab es Probleme und wie viel Zeit hat Sie das gekostet?

[jetzt kommt's]

Die 10 Gebote (gültig für alle Übungsblätter)

1. Im Laufe der Veranstaltung steigen sowohl die Anforderungen, als auch die Menge der Konstrukute, die Sie für Ihre Programme benutzen dürfen. Als Faustregel gilt: Sie dürfen grundsätzlich alles benutzen, was bereits dran war, und wenn bei einem Übungsblatt eine Anforderung dazu kommt, gilt diese auch für alle zukünftigen Übungsblätter.

2. Die Designvorlage (.TIP Datei, falls vorhanden) ist nicht verbindlich. Sie beruht aber auf viel Erfahrung, von daher sollten Sie sich dreimal überlegen, wenn Sie davon abweichen.

3. Schreiben Sie Ihren Code modular: wenn ein Teil des Codes für sich alleine umfangreich bzw. komplex genug ist oder mehrfach benötigt wird, gehört er in eine eigene Funktion.

4. Dokumentieren Sie Ihren Code: jedes Stück Code, dessen Funktionsweise sich nicht unmittelbar duch Lesen des Codes ergibt, sollte erklärt werden. Jede nicht-triviale Funktion sollte erklärt werden. Ihre Erklärungen sollte kurz und aussagekräftig sein. Manchmal ist ein Beispiel nützlicher (und kürzer) als eine abstrakte Erklärung.

5. Schreiben Sie für jede nicht-triviale Funktion einen Unit Test. Jeder Unit Test sollte mindestens ein nicht-triviales Beispiel überprüfen. Wenn es kritische Grenzfälle gibt, die sich durch wenig Aufwand leicht nachprüfen lassen (z.B. Verhalten einer Methode bei leerem Eingabefeld), sollten Sie das ebenfalls tun.

6. Laden Sie Ihren Code vollständig in unser SVN hoch, inklusive *Makefile*. Andere Dateien (insbesondere *.o* Dateien, die wir ab der 2. Vorlesung kennenlernen) dürfen nicht mit hochgeladen werden, sonst gibt es Karmaminuspunkte oder Schlimmeres.

7. Die finale Version von Ihrem Code muss fehlerfrei auf Jenkins durchlaufen. Zwischenversionen (siehe die Erklärung zum SVN in Aufgabe 4) müssen nicht auf Jenkins durchlaufen.

8. Code, der auf Jenkins nicht kompiliert („make compile“) oder der keine Tests hat, wird nicht korrigiert (= 0 Punkte), weil das unzumutbar viel Arbeit für die Tutoren wäre.

9. Wenn Sie ein Problem bei der Implementierung haben, suchen Sie ein paar Minuten (auf Google oder auch auf dem Forum, da wurden sehr viele Fragen schon mal gestellt) nach dem Fehler. Wenn Sie nicht fündig werden, fragen Sie gerne auf dem Forum, da wird Ihnen in der Regel schnell geholfen. Eine Anleitung für das richtige Fragen auf dem Forum findet sich auf dem Wiki (Stichwort: konkret fragen mit Fehlermeldung + Zeilennummer + relevantem Code, und nicht nur „Mein Code geht nicht“, sonst kann Ihnen keiner helfen).

10. Sie können gerne zusammen über die Übungsblätter nachdenken, aber der Code bzw. die Lösungen müssen zu **100%** selber geschrieben werden. Auch das teilweise Übernehmen gilt als Täuschungsversuch, mit den entsprechenden Konsequenzen. Wir müssen das so deutlich sagen, weil es in der Vergangenheit immer wieder vorgekommen ist. Man lernt nichts, wenn man Code bzw. Lösungen von anderen übernimmt und es ist auch einfach unfair gegenüber dem Großteil der Teilnehmer, die sich ehrlich Mühe geben.