Ludwig-Maximilians-Universität München
Mathematisches Institut
Gregor Kleen

Programmieren II für Studierende der Mathematik

Blatt 11

Aufgabe 12 Ziel ist es die Lösung der vorhergegangenen Übungsaufgabe von der Äquivalenz von p-Normen in eine bessere Code-Struktur zu bringen und mit unittests zu versehen. Falls Sie das entsprechende Übungsblatt nicht (vollständig) bearbeitet haben, können Sie auch den bereitgestellten Lösungsvorschlag verwenden. Bei der Umstrukturierung von bestehendem Code ohne (wesentliche) Änderung der Funktionalität spricht man von *refactoring*.

Legen Sie ein Verzeichnis für Ihr Projekt an. Verschieben Sie Ihre Lösung in eine Quellcodedatei (z.B. und im Folgenden `pnorms.cpp`) unterhalb des von Ihnen erstellten Verzeichnisses. Formulieren Sie eine Datei `meson.build` um die Übersetzungseinheit `pnorms.cpp` zu einer ausführbaren Binärdatei zu übersetzen und legen Sie diese ebenfalls in dem Verzeichnis ab.

Sie sollten nun in der Lage sein mit einem geeigneten `meson`-Befehl das builddir anlegen zu lassen, ihre Lösung mit `ninja` zu übersetzen und Sie danach interaktiv auszuführen.

Erstellen Sie eine Datei `pnorms.h` mit Deklarationen für die in `pnorms.cpp` definierten templates. Inkludieren Sie `pnorms.h` auch in `pnorms.cpp`.

Hinweis. Sie werden die Definitionen, die Teil der Klasse `complex_norm` sind, in `pnorms.cpp` ändern müssen. Sie dürfen die Klasse `complex_norm` nicht erneut vereinbaren und müssen stattdessen ihre Komponenten außerhalb einer Klassen-Deklaration definieren.

An dieser Stelle sollten Sie erneut in der Lage sein ihre Lösung zu übersetzen und interaktiv auszuführen.

Deklarieren Sie in der Header-Datei und definieren Sie in `pnorms.cpp` eine zusätzliches Funktionstemplate `pnorm_bound`. Die Funktionen sollen einen Parameter z vom Typ `complex<`T`>` akzeptieren und einen Wert vom Typ T zurückgeben. Zusätzlich zu T soll das template zwei weitere Parameter p_1 und p_2 akzeptieren. Der zurückgegebene Wert soll $\frac{\|z\|_{p_1}}{\|z\|_{p_2}}$ sein.

Passen Sie Ihr Hauptprogramm in `pnorms.cpp` an, sodass dieses `pnorm_bound` geeignet verwendet, wo immer sinnvoll möglich.

Lagern Sie ihr Hautprogramm aus in eine zweite Übersetzungseinheit `pnorms_interact.cpp`.

Hinweis. Es wird notwendig sein die in `pnorms_interact.cpp` verwendeten Instanzen der templates aus `pnorms.h` in `pnorms.cpp` explizit zu instanziieren.

Sie werden zudem Ihre `meson.build` anpassen müssen, sodass `pnorms.cpp` nun als Programmbibliothek übersetzt und mit dem Resultat der Übersetzung von `pnorms_interact.cpp` gelinkt wird.

An dieser Stelle sollten Sie erneut in der Lage sein ihre Lösung zu übersetzen und interaktiv auszuführen.

Fügen Sie Ihrem meson Projekt das unit testing framework Google Test als subproject hinzu. Lassen Sie es von meson aus der WrapDB installieren.

Fügen Sie eine weitere Übersetzungseinheit `pnorms_unittests.cpp` hinzu. Spezifizieren Sie in `meson.build` geeignet, dass das Ergebnis der Übersetzung von `pnorms_unittsets.cpp` zu einer ausführbaren Binärdatei mit dem Befehl `meson test` ausführbar sein soll. Geben Sie hierbei auch an, dass die Binärdatei gelinkt werden soll mit der in der meson Variable `gtest_main_dep` im subproject `gtest` bereitgestellten dependency.

Hinweis. Es ist dann nicht notwendig in `pnorms_unittests.cpp` ein Hauptprogramm zu implementieren.

Selbst mit `pnorms_unittests.cpp` einer komplett leeren Datei sollten Sie Ihr Projekt weiterhin übersetzen und interaktiv ausführen können. Es sollte auch möglich sein mit einem geeignete `meson test`-Befehl die aus `pnorms_unittests.cpp` übersetzte ausführbare Binärdatei ausführen zu lassen. Es werden hierbei aber natürlich noch keinerlei Tests ausgeführt.

Implementieren Sie in `pnorms_unittests.cpp` beliebig viele unit tests, die Ihnen sinnvoll erscheinen.

Es sei im Folgenden $B(z) := \frac{\|z\|_2}{\|z\|_3}$; Implementieren Sie aber mindestens auch eine Test-Suite `PNorms` bestehend aus den folgenden Tests:

- $B(1+i) \approx 1{,}122\,462\,048\,309$

  Hinweis. Verwenden Sie das von Google Test bereitgestellte Präprozessor-Makro `EXPECT_NEAR(`a`,` a'`,` ε`)`, welches prüft, dass gilt: $|a - a'| \leq \varepsilon$.

  ε soll hierbei nicht unnötig groß sein.

- $B(1{,}5 + 2{,}0i) \leq B(1+i)$

  Hinweis. Verwenden Sie hier und im Folgenden das von Google Test bereitgestellte Präprozessor-Makro `EXPECT_LE` (analog zu `EXPECT_EQ`).

- $B(7{,}0 + 7{,}0i) \leq B(1+i)$

- $B(3{,}7 + 42{,}1i) \leq B(1+i)$