Institut für Numerische und Angewandte Mathematik
FB Mathematik und Informatik der Universität Münster
Prof. Dr. Mario Ohlberger
Tobias Leibner

18. September 2019

Übung zum Kompaktkurs

Einführung in die Programmierung mit C++

Sommersemesterferien 2019 — Blatt 3

Aufgabe 1 (Die `Vector` Klasse)

Vervollständigen Sie die `Vector` Klasse in der Datei `vector.hh`. Laden Sie sich dazu die Datei `test_vector.cc` von der Projekt-Homepage herunter. Kompilieren Sie diese Datei und ergänzen Sie alle Funktionalität in der Klasse `Vector`, die nötig ist (dies gilt vor allem für die bisher auskommentierten Zeilen). Insbesondere benötigen Sie

- einen `operator-`;
- einen `operator*`, der das Skalarprodukt von zwei Vektoren bildet;
- einen *default* Konstruktor, der es erlaubt, ein Objekt ohne Argumente anzulegen. Fügen Sie dazu einen weiteren Konstruktor ohne Argumente hinzu und initialisieren Sie die Member auf sinnvolle Art und Weise oder fügen Sie dem existierenden Konstruktor sinnvolle default Argumente hinzu;
- eine Methode `scal`, welche alle Einträge des Vektors mit einem `double alpha` skaliert;
- einen `operator*=`, welcher `scal` ausführt und eine Referenz auf das Objekt selber zurück gibt;
- eine Methode `axpy`, welche einen `Vector u` für einen übergebenen `Vector v` und einen `double alpha` auf

  $$u := \alpha v + u$$

  setzt.

Achten Sie insbesondere darauf,

- dass alle Methoden, welche ein Objekt nicht ändern, mit `const` markiert werden;
- wann immer Möglich die Korrektheit von Indizes mit `assert` zu überprüfen.

Aufgabe 2 (Die `Matrix` Klasse)

Legen Sie eine neue Klasse `Matrix` in einem neuen Header `matrix.hh` an und implementieren Sie sinnvolle Member und Methoden. Laden Sie sich dazu die Datei `test_matrix.cc` von der Projekt-Homepage herunter. Kompilieren Sie diese Datei und ergänzen Sie alle Funktionalität in der Klasse `Matrix`, die nötig ist; gehen Sie dazu wie bei der Klasse `Vector` vor.
Hinweise:

- Nutzen Sie, wann immer möglich, die Funktionalität von `Vector` aus.
- Ein sinnvoller Konstruktor für eine Matrix ist durch `Matrix(unsigned int rr, unsigned int cc, double value)` gegeben.
- Sinnvolle Member für eine Matrix sind die nicht negativen Zahlen `rows_` und `cols_`, als Speicherlayout bietet es sich an, je eine Zeile der Matrix als ein Objekt vom Typ `Vector` aufzufassen. Sie benötigen also einen Vektor von `rows_` vielen `Vector`en der Länge `cols_` (was Sie mit Hilfe von `std::vector<Vector>` erreichen können).
- Die Ausgabe einer Matrix

  $$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

  soll folgendermaßen aussehen:
  ```
  [1 2 3;
   4 5 6]
  ```
- Es soll zwei `operator*` geben, einmal zur Multiplikation einer `Matrix` mit einem `Vector`, einmal zur Multiplikation zweier Matrizen. Achten Sie in beiden fällen auf passende Größen.
- (Bonusaufgabe) Zur Berechnung der Determinante einer Matrix bietet sich zum Beispiel der Laplacesche Entwicklungssatz an.