Übungen zur Vorlesung

"Objektorientiertes Programmieren im Wissenschaftlichen Rechnen"

Dr. O. Ippisch, Dr. C. Engwer

Abgabe am 06. 07. 2010 in der Vorlesung

ÜBUNG 1 TYPE TRAITS FÜR MATRIZEN

In Übung 8, Aufgabe 3 wurde die Frobenius Norm für Matrizen implementiert. Hierbei wurde als Rückgabewert immer `double` verwendet.

Wir wollen nun die Normberechnung so erweitern, dass sie für sehr verschiedene Zahlentypen funktioniert. Dies wollen wir mit Hilfe des Type Traits Konzeptes realisieren. Es gibt ein paar Dinge zu beachten:

- Der Rückgabewert hängt zunächst vom Zahlentype der Matrix ab.
- Die Normberechnung soll auch für `complex<T>` und `Rational` (siehe Übungsblatt 1) funktionieren.
- Der Rückgabewert ist nicht immer der Zahltyp der Matrix, z.B. bei komplexen Zahlen ist es der zugrundeliegende reelle Typ.
- Auch die Berechnung des Betragsquadrats ist nicht für alle Datentypen gleich.

Überlegen / Programmieren sie:

1. Welche Variabilitäten sind im Algorithmus notwendig?
2. Welche Traits möchten Sie dafür einführen?
3. Entwerfen sie Testprobleme für Matrizen mit ganze, rellen, rationalen und komplexen Zahlen (3×3 Matrizen reichen da völlig).
4. Implementieren sie die entsprechen Traits Klassen und passen sie die `frobeniusnorm` Funktion an. Schreiben sie Spezialisierungen der Traits, so dass die Normberechnung für `double`, `float`, `int`, `Rational`, `complex<T>` funktioniert.
5. Testen sie ihr Programm anhand der Testprobleme für die Typen `double`, `float`, `int`, `Rational`, `complex<double>`, `complex<float>`, `complex<int>` und `complex<Rational>`.

10 Punkte

ÜBUNG 2 MATRIXNORM VIA POLICY

Je nach Anwendung möchte man zwischen verschiedenen Matrixnormen auswählen können.

Es sei $A \in \mathbb{R}^{n \times n}$. Wir betrachten folgende Matrixnormen:

- Frobeniusnorm

$$\|A\|_{\mathrm{F}} = \sqrt{\sum_{i,j} |a_{ij}|^2},$$

- Zeilensummennorm
$$\|A\|_\infty = \max_i \sum_j |a_{ij}|,$$
- Gesamtnorm
$$\|A\|_\mathrm{G} = n \cdot \max_{i,j} |a_{ij}|$$

Schreiben sie eine Funktion

```
template<class P=FrobeniusNorm, class M>
MatrixTraits<M>::realType matrixNorm (const M & A);
```

welche mit einer Policyklasse `P` parametrisiert wird und dadürch in der Lage ist alle drei Normen zu berechnen.

Man diese Normen in einer gemeinsamen Art formulieren, so dass man über alle Zeilen iteriert, für jede Zeile iteriert man über die Spalten und führt eine Operation `OPentry` für jeden Eintrag aus. Für jede Zeile wird dann ein `OProw` ausgeführt...
Pseudocode:

```
matrixNorm<P> (Matrix A) {
    result = 0;
    foreach (row r of A) {
        row_result = 0;
        foreach (entry x of r) {
            row_result = P::OPentry(x, row_result);
        }
        result = P::OProw(row_result, result);
    }
    return result;
}
```

Die Policyklasse `P` soll alle notwendigen Operatoren und Information mitliefern, die nötig sind, um die Norm vollständig zu berechnen.

1. Welche Informationen / Operatoren muss die Policyklasse mitliefern?
2. Wie sehen diese Informationen für die drei Normen aus?
3. Welche Templateparameter (wenn überhaupt) brauchen die Policyklassen?
4. Implementieren sie die generische `matrixNorm` Funktion.
5. Implementieren sie die Policyklassen für alle 3 Normen.
6. Testen sie die Klassen mit folgendem Beispiel:

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 3 & 2 \\ 0 & 1 & -4 \end{pmatrix}, \qquad \|A\|_\mathrm{F} = 6, \qquad \|A\|_\infty = 6, \qquad \|A\|_\mathrm{G} = 12$$

10 Punkte

Wie immer gilt: Kommentieren Sie Ihr Programm. Erklären Sie was Sie tuen.