Übungen zur Vorlesung

"Objektorientiertes Programmieren im Wissenschaftlichen Rechnen"

Dr. O. Ippisch, Dr. C. Engwer

Abgabe am 8. 06. 2010 in der Vorlesung

ÜBUNG 1 MATRIZEN UND TEMPLATES

In der Vorlesung wurden Templates als Technik der generischen Programmierung vorgestellt. Sie erlauben es Algorithmen unabhängig von zugrundeliegenden Datenstrukturen zu entwickeln.

In dieser Übung werden wir eine existierende Implementierung einer Matrix Klasse zu einer Template Klasse erweitern. Matrizen werden immer wieder in numerischer Software benötigt. In der Vorlesung wurde bereits eine Implementierung für den Zahlentyp `double` vorgestellt, auf der Vorlesungs-Homepage finden sie die Klasse `MatrixClass`. Je nach Anwendung möchte man jedoch Matrizen basierend auf andere Zahlentypen, z.B. `complex`, codefloat oder `int`, haben.

Durch die Verwendung von Templates soll redundanter Code vermieden werden, so dass man eine Implementierung von `MatrixClass` hat, die für alle verschiedenen Typen verwendet werden kann.

MatrixClass für `double` — Grundlegende Eigenschaften:

- Matrix Einträge werden als Vektor von Vektoren gespeichert:
  ```
  vector<vector<double> > a_;
  ```
- Der Runde-Klammer-Operator gewährt Zugriff auf einzelne Einträge:
  ```
  double &operator()(int i, int j);
  double operator()(int i, int j) const;
  ```
- Die Matrix bietet die arithmetischen Operationen Multiplikation und Addition:
  ```
  MatrixClass &operator*=(double x);
  MatrixClass &operator+=(const MatrixClass &b);
  ```

Zusätzlich gibt es verschiedene freien Funktionen, die arithmetische Operationen basierend auf $*=$ und $+=$ implementieren:

- `operator*(const MatrixClass &a, double x);`
- `operator*(double x,const MatrixClass &a);`
- `operator+(const MatrixClass &a, const MatrixClass &b);`

Die aktuelle Version finden sie auf der Vorlesungs-Homepage in den Dateien `matrix_double.h` und `matrix_double.cc`. Zusätzlich finden sie hier ein Testprogramm `test_matrix_double.cc`.

Aufgaben:

1. **Template-Klassen und -Funktionen:** Ändern sie die bereitgestellte Implementierung von `MatrixClass` in eine Template-Klasse, so dass diese für verschiedene Zahlentypen verwendet werden kann. Passen sie die main-Funktion des Testprogramms an, so dass die Template-Varianten getestet werden.

   Hinweis: Auch die freien Funktionen

- `operator*(const MatrixClass &a, double x)`
- `operator*(double x,const MatrixClass &a)`
- `operator+(const MatrixClass &a, const MatrixClass &b)`

müssen angepasst werden.

2. Teilen sie die templatisierte Klasse `MatrixClass` auf zwei Klassen auf:
   Split up the templated `MatrixClass` into two classes:

   - eine Klasse `MatrixClass`, ohne numerische Operatoren. Diese repräsentiert eine Matrix von Zahleinträgen und hat folgende Methoden:
     - Konstructor (wie in `MatrixClass`)
     - `operator()(int i, int j)` für den Zugriff auf einzelne Einträge
     - `Resize` und `print` Methoden (wie in `MatrixClass`)
   - eine numerische Matrix Klasse `NumMatrixClass`, die von `MatrixClass` erbt. Die numerische Matrix Klasse erlaubt zusätzlich arithmetische Operationen auf der Matrix (sie erster Teil).

   Passen sie die auch die freien Funktionen und die main-Funktion des Testprogramms entsprechend an.

14 Punkte

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