Dipl.-Math. Christian Himpe, Dipl.-Math. Felix Schindler WS 13/14

Praktikum:

Einführung in die Programmierung zur Numerik mit C++

Mittwoch, 19.03.2014

Aufgabe 1 (Matrix Klasse)
Schreiben Sie eine Matrix-Klasse mit deren Hilfe Matrix-Objekte beliebiger Größe angelegt werden können. Die Matrizen sollen sich addieren, multiplizieren, transponieren und skalieren lassen. gehen Sie dabei schrittweise wie folgt vor (und testen Sie jeden Schritt bevor sie zum Nächsten übergehen):

1. Legen Sie eine Header-Datei `matrix.hh` an, in der die Matrix-Klasse definiert wird.
2. Legen Sie eine Datei `matrix_main.cc` an, die den Header `matrix.hh` einbindet. `matrix_main.cc` soll zum Testen des Codes verwendet werden.
3. Legen Sie eine Matrix-Klasse ohne Eigenschaften an.
4. Legen Sie innerhalb der Matrix-Klasse Variablen für die Spalten- und Zeilenanzahl an. Fügen Sie Methoden hinzu, mit deren Hilfe beide Werte abgefragt werden können.
5. Legen sie einen dynamischen Array `matrixEintrag` vom Typ `double**` an.
6. Schreiben Sie einen Konstruktor für ein Matrix-Objekt, mit dem sich eine Matrix mit m-Zeilen und n-Spalten anlegen lässt: `Matrix( int m, int n )`. Beachten Sie, dass Spalten- und Zeilenanzahl zur Compile-Zeit möglicherweise noch nicht feststehen und dass der Speicherplatz deshalb dynamisch beschafft werden muss. Alle Matrixeinträge sollen mit 0 initialisiert werden.
7. Erweitern Sie den Konstruktor so, dass auch eine Einheitsmatrix angelegt werden kann.
8. Fügen Sie der Klasse eine Methode `eintrag(...)` zum Abfragen und eine Methode `eintrag(...)` zum Ändern von Matrixeinträgen hinzu.
9. Fügen Sie der Klasse eine Methode `print` hinzu, deren Aufruf die erzeugte Matrix ausgibt.
10. Schreiben Sie einen Kopier-Konstruktor für Matrix-Objekte:
    `Matrix( const Matrix& originalMatrix )`.
11. Fügen Sie der Klasse eine Methode namens `skaliere` hinzu, welche ein Objekt der Matrix-Klasse mit einem `double` multipliziert.
12. Fügen Sie der Klasse eine Methode namens `groesseAendern` hinzu, mit dessen Hilfe sich Spalten- und Zeilenanzahl eines bereits angelegten Matrix-Objekts nachträglich ändern ässt.
13. Fügen Sie der Klasse eine Methode namens `transponiere` hinzu, welche die Matrix transponiert.
14. Fügen Sie der Klasse eine Methode namens `skalarprodukt` hinzu, welche das Ska-

larprodukt von zwei $n \times 1$-Matrizen bildet.

15. Fügen Sie der Klasse die Operatoren $+=$ und $*=$ zur Addition und Multiplikation von Matrizen hinzu.
16. Fügen Sie der Klasse eine Methode namens `spur` hinzu, welche die Spur der Matrix berechnet.
17. Überladen Sie die Methode `spur` derart, dass eine Matrix übergeben werden kann und die Spur des Produkts der Matrizen EFFIZIENT berechnet wird.
18. Testen Sie Ihr Programm, indem sie das 9-fache von $\begin{pmatrix} 1 & 4 & 7 & 4 \\ 2 & 6 & 7 & 3 \\ 8 & 3 & 5 & 1 \\ 4 & 8 & 7 & 9 \end{pmatrix}$ mit der Transponierten von $\begin{pmatrix} 3 & 1 & 7 & 3 \\ 8 & 3 & 4 & 5 \end{pmatrix}$ multiplizieren.

Zusatzaufgabe (Destruktor für Matrix-Objekte)
Wenn Speicherplatz dynamisch angefordert wurde, muss dieser selbsständigt wieder frei gegeben werden. Da Matrix-Objekte dynamisch Speicher für ihre Einträge anforden, muss hierfür ein sogenannter Destruktor definiert werden, der vorschreibt, wie Matrix-Objekte wieder zerstört werden können, sobald sie nicht mehr benötigt werden. Der Destruktor wird selbstständig vom Programm aufgerufen. Schreiben Sie einen Destruktor für die Matrix-Klasse. Die Syntax ist:

```
~Matrix()
  {
    ... //Speicher für Einträge wird freigegeben
  }
```