Prof. Dr. M. Ohlberger, P. Henning WS 2010/2011

Praktikum:

Einführung in die Programmierung zur Numerik mit C++

Donnerstag, 17.03.2011

Aufgabe 1 (Vererbung)
Leiten Sie von der Klasse `Matrix` die Klasse `Diagonalmatrix` ab. Gehen Sie folgendermaßen vor:

1. Schreiben Sie einen geeigneten Konstruktor und überladen Sie diesen (beispielsweise mit einem zusätzlichen leeren Konstruktor).
2. Redefinieren Sie die Funktionen `welcherWert()` und `setzeWert()`, so dass diese jeweils nur eine Positionsangabe übernehmen (die Nummer des Diagonaleintrags).
3. Redefinieren Sie die Funktionen `addiere()` und `multipliziere()`. Die Addition und Multiplikationen von Diagonalmatrizen kann einfacher implementiert werden als die von allgemeinen Matrizen (in der Berechnung spart dies Rechenaufwand).
4. Redefinieren Sie die Funktion `print()`, so dass nur die Diagonalelemente ausgegeben werden.
5. Schreiben Sie eine neue Funktion, die die Determinante einer Diagonalmatrix ausrechnet.
6. Testen Sie Ihr Programm, indem Sie die Matrizen $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 5 \end{pmatrix}$ und $\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$ addieren, multiplizieren und sich von den entstandenen Matrizen die Determinanten ausgeben lassen.

Aufgabe 2 (SOR-Verfahren)
Implementieren Sie das SOR-Verfahren für $\omega = 1, 1.2, 1.4, 1.6, 1.8$. Greifen Sie hierbei auf die Implementierungen von `matrix.h` und `matrix.cc` zurück. Testen Sie Ihr Programm: Suchen Sie die Lösung x für das Gleichungssystem $Ax = b$, mit

$$\mathrm{A}=\begin{pmatrix} 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 4 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 \end{pmatrix}, \quad \mathrm{b}=\begin{pmatrix} 0.001 \\ 0.001 \\ 0.001 \\ 0.001 \\ 0.001 \\ 0.001 \\ 0.001 \\ 0.001 \\ 0.001 \\ 0.001 \\ 0.001 \\ 0.001 \end{pmatrix}$$

Nehmen Sie den Einservektor als Startvektor und geben Sie sich eine maximale Anzahl von Iterationen vor. Z.B. `int maxIteration = 60;`.

Zur Erinnerung: Die Lösung wird mit dem SOR-Verfahren folgendermaßen berechnet:

$$x_i^{k+1} = (1-\omega)x_i^k + \omega \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{k+1} - \sum_{j=i+1}^{n} a_{i,j} x_j^k \right) / a_{i,i}$$

Hinweis:
Sei n die Matrixgröße von A ($n \times n$-Matrix), dann fassen Sie b und x^k als $1 \times n$-Matrizen auf, d.h.

```
unsigned int n = 12;
Matrix A(n,n);
Matrix b(n,1);
Matrix x(n,1);
```

Anschließend werden die jeweiligen Einträge mit der Methode `setzeWert` gesetzt. Verwenden Sie hierfür Schleifen und lassen Sie sich zur Kontrolle A und b mittels `print` ausgeben.