BERGISCHE UNIVERSITÄT
GESAMTHOCHSCHULE WUPPERTAL
GAUSS-STRASSE 20
42097 WUPPERTAL
(Korrespondenzanschrift)
42119 WUPPERTAL
(Lieferanschrift)
TELEX 8 592 262 bughw
TELEFAX (0202) 439-2901
TELEFON (0202) 439-1

Fachbereich 7

MATHEMATIK

Prof. Dr. Hans-Jürgen Buhl
Praktische Informatik / Numerik

e-mail: Juergen.Buhl@math.uni-wuppertal.de

Algorithmen und Datenstrukturen (Informatik II)

SS2001 – Übungsblatt 9

Abgabetermin: 2. Juli 2001

Aufgabe 1. *Matrizen, 3 Punkte*
In der folgenden Klasse `matrix`

```
////////////////////////////////
// Datei:   matrix.cc
// Version: 1
// Zweck:   matrix als Klasse
// Autor:   Hans-Juergen Buhl
// Datum:   10.02.99
////////////////////////////////

#include <iostream>
using namespace std;

class matrix{

  int low1;        // v(low1..high1, low2..high2)
  int high1;               // high1 >= low1
  int low2;
  int high2;               // high2 >= low2

  double* v;         // Startadresse fuer dyn. verwaltetes Exemplar

public:

  matrix(int h1, int l1, int h2, int l2, double d = 0.0);

  ~matrix(){ delete []v; };

  int lo1() const { return low1;  };
  int hi1() const { return high1; };
  int lo2() const { return low2;  };
  int hi2() const { return high2; };
```

```
  double& operator()(int i, int j);
  double  operator()(int i, int j) const;

};

matrix::matrix(int h1, int l1, int h2, int l2,
               double d) : low1(l1), high1(h1), low2(l2), high2(h2)
{
  int size1(h1-l1+1);
  if (size1 < 1) throw "falscher Matrix-Zeilenindex in Konstruktor";

  int size2(h2-l2+1);
  if (size2 < 1) throw "falscher Matrix-Spaltenindex in Konstruktor";

  v = new double[size1*size2];
  if (v == 0) throw "kein freier Speicherplatz mehr verfügbar";

  for (int j=l1; j <= h1; j++)
    for (int k=l2; k <= h2; k++)
      v[(j-l1)+(k-l2)*size1] = d;

  // oder:
  //
  // for (int i = 0; i < size1 * size2; i++)
  //   v[i] = d;

};

double& matrix::operator()(int i, int j)
{
  if ( (i < low1) || (i > high1)) throw "Indexverletzung bei Komponentenzugriff";
  if ( (j < low2) || (j > high2)) throw "Indexverletzung bei Komponentenzugriff";
  int size1(high1-low1+1);

  return v[(i-low1)+(j-low2)*size1];
};

double  matrix::operator()(int i, int j) const
{
  if ( (i < low1) || (i > high1)) throw "Indexverletzung bei Komponentenzugriff";
  if ( (j < low2) || (j > high2)) throw "Indexverletzung bei Komponentenzugriff";
  int size1(high1-low1+1);

  return v[(i-low1)+(j-low2)*size1];
};
int main()
{
  matrix m1(5, 2, 7, 0, 3.0);

  for (int i=m1.lo1(); i <= m1.hi1(); i++)
    for (int j=m1.lo2(); j <= m1.hi2(); j++)
      cout << i << "  " << j << " :       " << m1(i,j) << endl;
```

```
  return 0;
}
```

wird eine (high2-low2+1)-fache Zusammenfügung der gleich langen, mit gleichem Indexlaufbereich versehenen Spaltenvektoren durchgeführt.

Testen Sie das Programm

http://www.math.uni-wuppertal.de/~buhl/teach/exercises/Inf2-SS01/matrix.cc.

Wie ist es zu ändern, wenn die Matrix nicht — wie oben — spaltenweise, sondern zeilenweise abgespeichert werden soll? Testen Sie auch hier.

Aufgabe 2. *Ergänzungen in der Klasse matrix, 11 Punkte*
Konzipieren Sie die Ihnen nötig erscheinenden weiteren Konstruktoren (z.B. den Kopierkonstruktor, ...) und Modifikatoren (z.B.

`matrix operator=(const matrix&)`, ...

). Welche zusätzlichen Funktionen sind sinnvoll (z.B.

```
bool operator==(const matrix&) const,
friend ostream& operator<<(ostream& os, const matrix& m),
```

Zusammenfügen von Matrizen mit Matrizen

`friend matrix append(const matrix&, const matrix&)`,

die Frobeniusnorm

`friend double Norm(const matrix&)`,

die Transposition einer Matrix, die Addition zweier Matrizen, ...)? Schreiben und testen Sie diese Operationen. Beachten Sie dabei insbesondere, wann `delete []v` nötig ist, daß auch in Methoden der sichere Komponentenzugriff durch den operator `()` verwendet werden sollte, ...

Aufgabe 3. *matrix und vektor, 6 Punkte*
Schreiben Sie Multiplikationsoperatoren

```
matrix operator*(const matrix&, const matrix&)
vektor operator*(const matrix&, const vektor&)
matrix operator||(const vektor& x, const vektor& y)  // x·y^t
```

und testen Sie. Dokumentieren Sie dabei gut die nötigen Kompatibilitätsregeln und benutzen Sie in der Implementierung bei deren Verletzung Exceptions.