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

Informatik I

WS98/99 – Übungsblatt 12

Bearbeitungstermin: 5. Kalenderwoche

Aufgabe 1. *Vektoren als Klasse*

Testen Sie das folgende Programm:

```
//////////////////////////////////
// Datei:   vektor1.cc
// Version: 1.0
// Zweck:   Vektor als Klasse
// Autor:   Hans-Juergen Buhl
// Datum:   26.01.99
//////////////////////////////////

#include <iostream>

//using namespace std;

class vektor{

  const int low;        // v(low..high)
  const int high;

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

public:

  vektor(const int h, const int l = 1, const double d = 0.0);
                                             //  v(l..h) = d

  vektor(const double x[], int n);           // v(1..n) = x[0..n-1]

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

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



```
  int lo() const { return low;  };
  int hi() const { return high; };

  friend ostream& operator<<(ostream& os, const vektor& v);

};

vektor::vektor(const int h, const int l, const double d) : low(l), high(h)
{
  int size(h-l+1);
  if (size < 1) throw "falsche Vektor-Länge in Konstruktor";
  v = new double[size];
  if (v == 0) throw "kein freier Speicherplatz mehr verfügbar";
  for (int j=0; j < size; j++)
    v[j] = d;
};

vektor::vektor(const double x[], int n) : low(1), high(n)
{
  if (n < 1) throw "falsche Vektor-Länge in Konstruktor";
  v = new double[n];
  if (v == 0) throw "kein freier Speicherplatz mehr verfügbar";
  for (int i=0; i < n; i++)
    v[i] = x[i];
};

double& vektor::operator()(const int i)
{
  if ( (i < low) || (i > high)) throw "Indexverletzung bei Komponentenzugriff";
  return v[i-low];
};

double  vektor::operator()(const int i) const
{
  if ( (i < low) || (i > high)) throw "Indexverletzung bei Komponentenzugriff";
  return v[i-low];
};

ostream& operator<<(ostream& os, const vektor& w)
{
  os << "( ";
  os << w(w.lo());
  for (int i=w.lo()+1; i <= w.hi(); i++)
    os << " , " << w(i);
  os << " )";
  return os;
};

int main()
```

```
{
  vektor y(5, 1, 3.0);
  cout << y.lo() << " " << y.hi() << endl;
  cout << y  << endl;
  for (int i=y.lo(); i <= y.hi(); i++)
    y(i) = i*2;
  cout << y  << endl;

  vektor x(8, 2);
  cout << x.lo() << " " << x.hi() << endl;
  cout << x  << endl;
  for (int k=x.lo(); k <= x.hi(); k++)
    x(k) = k*2;
  cout << x << endl;

  double zh[] = { 1.0, 3.0, 2.0, 4.0 };
  vektor z(zh, 4);
  cout << z.lo() << " " << z.hi() << endl;
  for (int j=z.lo(); j <= z.hi(); j++)
    z(j) *= z(j);
  cout << z << endl;

  return 0;
}
```

Aufgabe 2. *Vektoren als Klasse, Forts.: Diskussion vektor1.cc*

- Welche verschiedenen Signaturen werden durch die Vereinbarung
  `vektor(const int h, const int l = 1, const double d = 0.0);`
  dem Benutzer zur Verfügung gestellt? Testen Sie!
- Schreiben Sie einen Konstruktor
  `vektor(const int h, const double d);`
  und testen Sie.
- Testen Sie auf Speicherlecks. Kommentieren Sie den Destruktor aus und testen Sie erneut auf Speicherlecks.
- Schreiben Sie eine Methode
  `double Norm() const;`
  zur Berechung der euklidischen Norm eines Vektors und testen Sie.
- Testen Sie den Zugriff auf eine nicht existierende Komponente eines Vektors.
- Warum ist die Vereinbarung
  `double operator()(const int i) const;`
  notwendig? Entfernen Sie dazu zunächst diese Methode und compilieren Sie erneut. Welche Regel sollten Sie also in Ihre `C++` Kladde aufnehmen?

Aufgabe 3. *Vektoren als Klasse, Forts.: Kopierkonstruktor*

Testen Sie:

```
...
  vektor(const vektor& w);
...
vektor::vektor(const vektor& w) : low(w.low), high(w.high)
{
  int size(high-low+1);
  v = new double[size];
  if (v == 0) throw "kein freier Speicherplatz mehr verfügbar";
  for (int j=0; j < size; j++)
    v[j] = w.v[j];
};
```

Was kann mit diesem Konstruktor erreicht werden?

Aufgabe 4. *Vektoren als Klasse, Forts.: Zuweisungsoperator*

Testen Sie

```
...
  int low; // v(low..high)
  int high;
...
  vektor& operator=(const vektor& w);
...
vektor& vektor::operator=(const vektor& w)
{
  if (this != &w)
  {
     int my_size(high-low+1);
     int w_size(w.high-w.low+1);

     if(my_size != w_size)
     {
        delete []v;
        v = new double[w_size];
        if (v == 0) throw "kein freier Speicherplatz mehr verfügbar";
     };

     high = w.high; low = w.low;
     for (int i=0; i < w_size; i++)
        v[i] = w.v[i];
  };
  return *this;
};
```

durch geeignete Aufrufe in `main`. Schreiben Sie eine Funktion
`static vektor ei(const int n, const int i);`
zur Rückgabe des Einheitsvektors $e_i \in \mathbb{R}^n$ und benutzen Sie diese.