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 13

Bearbeitungstermin: 6. Kalenderwoche

Aufgabe 1. *Vektoren als Klasse, Forts.: operator==*

Vergleichen Sie:

```
...
#include <bool.h>
...
   bool operator==(const vektor&w) const;
...
bool vektor::operator==(const vektor&w) const
{
    if ((high-low) != (w.hi()-w.lo())) return false;

    int i_l(low);
    int i_r(w.lo());
    for(; i_l <= high; i_l++, i_r++)
         if (v[i_l-low] != w(i_r))
            return false;

    return true;
};
```

und

```
...
#include <bool.h>
...
   bool operator==(const vektor&w) const;
...
bool vektor::operator==(const vektor&w) const
{
    if ((high-low) != (w.hi()-w.lo())) return false;

    int i_l(low);
    int i_r(w.lo());
```



```
    for(; i_l <= high; i_l++, i_r++)
        if ((*this)(i_l) != w(i_r))
           return false;

    return true;
};
```

Testen Sie dazu alle Pfade mit geeigneten Testbeispielen. Schreiben Sie eine Methode `operator!=(const vektor& w) const`, die den Operator `==` benutzt.

Ändern Sie die Methode `vektor& operator=(const vektor& w)` analog so ab, daß statt mittels des unsicheren Komponentenzugriffs durch `v[i]` die sicherere Komponentenselektion mittels `w(i)` benutzt wird.

Aufgabe 2. *Vektoren als Klasse, Forts.: operator+ und operator**

Schreiben Sie analog Methoden `vektor operator+(const vektor& w) const` und (friend-)Funktionen `vektor operator*(const vektor& w, double a)` sowie `vektor operator*(double a, const vektor& w)`.

Testen Sie geeignet und dokumentieren Sie Ihre Testplanung und -durchführung. Warum ist hier im Gegensatz zur Klasse `comp` eine doppelte Version von `operator*` erforderlich?

Schreiben Sie statt der Methode `double Norm() const` eine (friend-)Funktion `double Norm2(const vektor& w)` und diskutieren Sie Vor- und Nachteile.

Sind hier wirklich friend-Funktionen nötig?

Aufgabe 3. *Vektoren als Klasse, Forts.: operator+*

Konzipieren und testen Sie friend-Funktionen

`vektor operator+(const vektor&w, double a)`

und

`vektor operator+(double a, const vektor&w)`,

die komponentenweise Addition mit einem Skalar realisieren sollen.

Warum ist hier nicht auch „`const double d`“ bzw. „`const double& d`“ nötig?

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

Konzipieren und testen Sie einen Operator „Skalarprodukt“.

Aufgabe 5. *Vorbereitung: Matrizen als Klasse, hier: Aneinanderfügen von Vektoren*

Überlegen Sie sich eine „Index-Abbildung“, die zwei im Bereich lo1 ... hi1 bzw. lo2 ... hi2 laufende Indizes in einen eindimensionalen Index 0 ... ??? abbildet, ohne daß Lücken oder Überschneidungen auftreten.