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 3

Bearbeitungstermin: 47. Kalenderwoche

Aufgabe 1. *int-wertige Vektoren der Dimension dim*

Geben Sie das folgende Programm ein, übersetzen Sie es, und bringen Sie es zur Ausführung:

```
#include <algorithm>
#include <iostream>


int main ()
{
  const dim(6);
  int numbers[dim] = { 1, 2, 3, 4, 5, 6 };
  int result[dim];

  partial_sum (numbers, numbers + dim, result);

  for (int i = 0; i < dim; i ++)
    cout << result[i] << ' ';

  cout << endl;

  return 0;
}
```

Was wird hier ausgeführt? Wie erzeugt man in C++ also Variablen, die Werte in int^{dim} annehmen sollen? Wie sehen Literale des Typs int^{dim} aus? Wie wird auf die Komponenten $result_0, \ldots result_{dim-1}$ zugegriffen? Wie weit erstreckt sich im Code die Lebensdauer des int-Objektexemplars i ?

Benutzen Sie bei der Beantwortung dieser Fragen zur Hilfe die zeilenweise Programmausführung im Debugger!

Aufgabe 2. *Summe der Komponenten eines Vektors*

Wenn lediglich die Summe aller Komponenten eines Vektors berechnet werden soll, ist der Aufruf der Bibliotheksfunktion *partial_sum* zu aufwendig.

Ändern Sie das Programm so ab, daß Sie in einer for-Zählschleife diese Summe explizit selbst berechnen, und testen Sie das modifizierte Programm.

Aufgabe 3. *Euklidische Norm eines Vektors*

Schreiben Sie ein ähnliches Programm, daß mittels einer for-Zählschleife und der Funktionen $double\ sqrt(double)$ sowie $double\ pow(double, int)$ (aus math.h) die euklidische Norm eines Vektors aus $double^{dim}$ berechnet und ausdruckt.

Aufgabe 4. *Eigene Funktionen*

Im folgenden Programm sehen Sie, wie Sie eigene Funktionen deklarieren können:

```
#include <iostream>
#include <assert.h>

double average(double v[], int dim)
{
  assert(dim > 0);
  double result(0);

  for (int i = 0; i < dim; i ++)
    result += v[i];

  return result/dim;
};

int main ()
{
  const dim(6);
  double numbers[dim] = { 1, 2, 3, 4, 5, 6 };

  cout <<  average(numbers, 6) << endl;

  return 0;
}
```

Was tut dieses Programm? Testen Sie mit obigem Vektor die Ergebnisse von $average(numbers, 6)$, $average(numbers, 3)$, $average(numbers, 0)$.

Nach Meldungen der Art

```
...
Abort (core dumped)
```

können die mittels der Anweisung `debugger prgname core` den Hauptspeicherauszug „core" genauer untersuchen (Programmzeile des Abbruchs, Stack Backtrace, dim display, ...).

Was passiert bei Aufruf von $average(numbers, 7)$? Raten Sie zunächst und testen Sie dann! Bitte vermerken Sie eine entsprechende Warnung in ihrer Kladde (vgl. letztes Übungsblatt).

Benutzen Sie *average* auch für Vektoren anderer Dimensionen.