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 7

Bearbeitungstermin: 51. Kalenderwoche

Aufgabe 1. *Turn-Around-Zeiten von Algorithmen*

Mit Hilfe des `time(0)`-Funktionsaufrufs (vgl. Aufgabe 4/Übungsblatt 5) kann man die Differenz der (Tages-)Uhrzeiten zweier Algorithmenstellen — die sogenannte Turn-Around-Zeit — in Sekunden messen:

```
//////////////////////////////////
// Datei:   funcProc.cc
// Version: 1.0
// Zweck:   Benutze Funktion als Prozedur
// Autor:   Hans-Juergen Buhl
// Datum:   17.09.1998
//////////////////////////////////

#include <iostream>

#include <cstdlib>
#include <math.h>
#include <time.h>
#include <unistd.h>

//using namespace std;

time_t DateTime()
{
  time_t walltime(time(0));

#ifndef NDEBUG
  cerr << walltime << " Sekunden seit dem 1.1.1970 0Uhr GMT" << endl;
#endif

  return walltime;
};

int main()
```



```
{
  time_t start(DateTime());
  // ...
  sleep(2);
  // ...
  double f(sin(1.4567));
  for (int i = 0; i<1000000; i++) f = sin(f);
  // ...
  time_t LaufZeit(DateTime()-start);
  cout << "Laufzeit: " << LaufZeit << endl;

  DateTime();

  return 0;
}
```

Zugleich demonstriert dieses Programm die Benutzung einer Funktion nicht in einem Ausdruck sondern als Anweisung.

Testen Sie obiges Programm. Vergleichen Sie die Ergebniszeit bei mindestens 5-maligem Start von `funcProc`. Versuchen Sie dabei auch während eines Programmlaufs durch das `ls`-Kommando intensiven Festplattenzugriff zu erzeugen. Interpretieren Sie!

Übersetzen Sie einmal mittels

`CC -o funcProc -g -I/opt/local/include funcProc.cc`

und dann mittels

`CC -o funcProc -DNDEBUG -I/opt/local/include funcProc.cc`

und führen Sie das erzeugte Programm `funcProc` je einmal aus. Welche Schlüsse ziehen Sie? Leiten Sie die Programmausgabe in eine Datei um. Benutzen Sie „`> AusgabeDatei`“ und „`>& AusgabeDatei`“ (beziehungsweise „`> AusgabeDatei`“ und „`> AusgabeDatei 2>&1`“, falls Sie nicht die `csh` benutzen). Was geschieht?

Aufgabe 2. *Klasse quadGleichung*

Ergänzen Sie die Klasse `quadGleichung` der Vorlesung um die Methode `solve()`, und führen Sie die `switch`-Anweisung im Hauptprogramm `main()` genauer aus.

Testen Sie Ihr vervollständigtes Programm mit geeigneten Koeffiziententripeln. Führen Sie dazu zuvor eine Testplanung durch, und dokumentieren Sie diese inklusive der zu erwartenden Ergebnisse.

Aufgabe 3. *Kleine Beispielfunktionen mit Schleifen und Fallunterscheidungen*

Schreiben Sie kleine Funktionen

a.) `int ggt(int, int)`

b.) `double abs(double)`

c.) `double min(double, double)`

f.) `int min(int, int)`

e.) `long factorial(long)`

f.) `double power(double, int)`

h.) `void sqrt_table(double start, double end, int anz_intervals)`

Planen Sie auch hier zuvor die nötigen Testfälle, und dokumentieren Sie diese inklusive der zu erwartenden Ergebnisse. Protokollieren Sie dann Ihre Tests (auch die eventuell fehlgeschlagenen!).

Benutzen Sie für die Teilaufgabe f.) nicht die Funktion `pow()` aus `math.h`, sondern berechnen Sie die Potenz mit Hilfe der Grundrechenarten selbst.