Informatik I für D-ITET Musterlösung 5 WS 04/05

URL: http://www.ti.inf.ethz.ch/ew/courses/inf1-ITET/

Aufgabe 1

Keine Abgabe erforderlich.

Aufgabe 2

Programm: math.h

```
// Programm: math.h
// Kleine Bibliothek mathematischer Funktionen.

namespace ifet {

  double sqrt(double n);
  // PRE: n >= 0.
  // POST: Gibt Approximation der Quadratwurzel von n zurueck.

  double random(double x);
  // POST: Gibt eine Pseudozufallszahl in [0,1) zum Startwert x zurueck.

} // namespace ifet
```

Programm: math.C

```
// Programm: math.C
// Kleine Bibliothek mathematischer Funktionen.

#include <IFET/math.h>
#include <cassert>
#include <limits>

namespace ifet {

  double sqrt(double n)
  {
    // PRE: n >= 0.
    // POST: Gibt Approximation der Quadratwurzel
    //       von n zurueck.

    assert(n >= 0);
    if (n == 0) return 0;

    // Newton-Iteration: x_{i+1} = 1/2 (x_i + n/x_i) Wir beginnen mit
    // dem Startwert x_0:=(n+1)/2. Da (n+1)/2-sqrt(n) =
    // (n-2sqrt(n)+1)/2 = (sqrt(n)-1)^2/2 > 0, gilt x_0 > sqrt(n).
    //
    // Wie in der Theorie verlangen wir, dass der Approximationswert
    // in jedem Schritt kleiner wird. Nur muss sich hier die Aenderung
    // auch innerhalb der double Mantisse auswirken.
    //
    // Damit terminiert das Programm, und wir koennen sogar beweisen,
```

```
    // dass "ungefaehr" das richtige herauskommt

    double curr = (n + 1) / 2; // Startwert
    double prev;               // voriger Wert

    const double roundoff =
      curr * std::numeric_limits<double>::epsilon();

    do {
      prev = curr;
      assert(curr > 0);
      curr = (curr + n / curr) / 2;
    } while (prev - curr > roundoff);

    return curr;

  } // double sqrt(double)

  double random(double x)
  {
    // POST: Gibt eine Pseudozufallszahl in [0,1) zum Startwert x
    // zurueck.

    // Lineare Kongruenzmethode zur Erzeugung von Pseudozufallszahlen
    // x_0, x_1, x_2, ...  mit Hilfe der Formel x_i = (a * x_(i-1) +
    // c) % m

    // Der Super-Duper Generator nach G. Marsaglia. The structure of
    // linear congruential sequences. In S. K. Zaremba, editor,
    // Applications of Number Theory to Numerical Analysis, pages
    // 248-285. Academic Press, New York, 1972.

    const double m = 4294967296.0; // Modulus, 2^32
    const double a = 69069.0;      // Multiplikator
    const double c = 1.0;          // Additiver Term

    // Da a < 2^17, d.h. (m-1)*a < 2^49, ist die Berechnung auf double
    // und int exakt.

    // Berechnung
    x = (x * m * a + c) / m;

    return x - int(x);

  } // double random(double)
} // namespace ifet
```

Aufgabe 3

Programm: muenzwurf.C

```
// Programm: muenzwurf.C
// berechnet eine Folge von Pseudozufallszahlen, simuliert damit
// Muenzwuerfe und ueberprueft deren Verteilung statistisch
```

```
#include <iostream>
#include <IFET/math.h>

int main()
{
  // lies Anzahl zu werfender Muenzen ein
  std::cout << "Anzahl von Muenzwuerfen: ";
  unsigned int anzahl;
  std::cin >> anzahl;

  // Erzeugung und Ausgabe
  std::cout << "Folge von " << anzahl << " Muenzwuerfen: ";
  unsigned int kopf = 0; // # "Kopf"-Wuerfe
  unsigned int zahl = 0; // # "Zahl"-Wuerfe

  double r = 0; // Startwert
  while (anzahl > 0) {
    // fuehrendes Bit der Zufallszahl
    if (r >= .5) {
      std::cout << "K";
      ++kopf;
    } else {
      std::cout << "Z";
      ++zahl;
    }
    r = ifet::random(r);
    --anzahl;
  }
  std::cout << "\nKopf: " << kopf << "-mal"
            << "\nZahl: " << zahl << "-mal" << std::endl;
  return 0;
}
```