Universität Duisburg-Essen
Fakultät für Mathematik
Dr. O. Rheinbach

SS11

2011

Aufgabe 14

Das folgende numerische Verfahren berechnet in wenigen Iterationen eine Näherung für $\sqrt{a}$. Es wird auch in Taschenrechnern eingesetzt.

```
Eingabe von a
x=1;
Solange |x^2-a|>10^{-7}
   x=(x+a/x)/2;
   Ausgabe von x
Ausgabe von x und x^2;
```

Programmieren Sie den Algorithmus in C++ und vergleichen Sie `x` mit der "exakten" Lösung. Die Anzahl der Dezimalstellen, die `cout` ausgibt, können Sie mit `cout.precision(14);` auf 14 Stellen ändern.

Kann es mit dem Programm in dieser Form irgendein Problem geben?

Aufgabe 15

Schreiben Sie ein C++-Programm, das eine Zahl n vom Benutzer einliest und durch Aufruf einer Funktion (nicht rekursiv) die Fakultät $n!$ berechnet.

In einem Fall soll der berechnete Wert von der Fakultäts-Funktion per Rückgabewert an `main` zurückgegeben werden:

```
int fakultaet(int n)
   {
// Programm
   }
```

Im anderen Fall soll der Rückgabewert durch Übergabe einer Referenz

```
void fakultaet(int n, int &ergebnis)
   {
// Programm
   }
```

erfolgen.

Aufgabe 16

Schreiben Sie eine rekursive Version der Fakulät `int fakultaet_rek(int)`, d.h. berechnen Sie `fakultaet_rek(n)` indem Sie `fakultaet_rek(n-1)` aufrufen. Verfolgen Sie die rekursiven Aufrufe (für ein kleines n) mit dem Debugger.

Aufgabe 17

Ergänzen Sie Aufgabe 20 um eine weitere Version der Fakultätsfunktion. Diese soll statt mit einer Referenz mittels eines Zeigers das Ergebnis an das Hauptprogramm zurückgeben.

Aufgabe 18

Schreiben Sie ein Programm, das vom Benutzer Zahlen einliest und sie hinten an den Vektor `v` hängt, wenn die eingebene Zahl positiv ist. Wenn sie negativ ist, wird die letzte Zahl vom Vektor entfernt. In jedem Schritt soll der komplette Inhalt des Vektors ausgeben werden. Bei Eingabe von 0 wird das Programm beendet, und es wird das Maximum, das Minimum, das arithmetische Mittel und das geometrische Mittel der Zahlen im Vektor ausgegeben. (Sie brauchen:`v.push_back(x); v.pop_back(); v[i]; v.size(); pow(x,y)`)

```
#include <iostream>
#include <vector>

using namespace std;


int main()
{
    vector<double> v;
    // hier Ihr Code
}
```

Bemerkungen: I. Die Länge eines Vektors kann außer mit den Methoden `push_back()` und `pop_back()` auch mit `resize()` verändert werden: Die Anweisung `v.resize(10);` setzt die Länge von `v` auf `10`. II. Wenn Sie aus anderen Programmiersprachen Arrays kennen sollten: Auch C++ hat Arrays, aber vermeiden Sie sie. Vektoren sind wesentlich flexibler, weniger fehleranfällig und fast ebenso effizient wie Arrays.

Aufgabe 19

Schreiben Sie eine einfache Matrix-Klasse. Das Grundgerüst für den Matrix-Datentyp soll so aussehen:

```
#include <iostream>
#include <vector>
```

```
using namespace std;

/* Anfang Deklaration der Klasse */
class Matrix
  {
   public:

   int m;    // Anzahl der Zeilen
   int n;    // Anzahl der Spalten
   vector<double> val;   // speichert die Werte

   /* hier fehlen die Deklarationen der Methoden */

  };
/* Ende Deklaration der Klasse */


   /* hier fehlen die Definitionen der Methoden */


int main()
  {
    // und so soll die Klasse verwendbar sein
    Matrix A;       // eine Matrix wird erschaffen;
    A.setsize(2,2); // die Matrix wird auf das Format 2x2 gebracht,
                    // alle Werte werden auf 0 gesetzt
    A.set(1,1,1.0); // a_11 wird auf 1.0 gesetzt;
                    // Fehlermeldung, falls die Indizes nicht zulaessig sind
    A.dump();       // gibt den Inhalt von A auf dem Bildschirm aus

    Matrix B;
    B.set(2,2,2.0); // b_22 wird auf 2.0 gesetzt
    B.dump();
  }
```

Aufgabe 20

Ergänzen Sie ihre Matrixklasse um eine Matrix-Matrix-Addition, d.h. folgendes Hauptprogramm soll möglich werden:

```
int main()
  {
    Matrix A;       // eine Matrix wird erschaffen;
    A.setsize(2,2); // die Matrix wird auf das Format 2x2 gebracht,
                    // alle Werte werden auf 0 gesetzt
```

```
    A.set(1,1,1.0);   // a_11 wird auf 1.0 gesetzt,
    A.dump();        // gibt den Inhalt von A auf dem Bildschirm aus

    Matrix B;
    B.setsize(2,2);
    B.set(2,2,2.0);   // b_22 wird auf 2.0 gesetzt

    A.add(B);        // A und B werden addiert und das Ergebnis steht in A,
                     // sofern die A,B die passenden Groessen haben,
                     // ansonsten Fehlermeldung

    A.sub(B);        // die Elemente von B werden von A abgezogen
                     // Ergebnis steht in A
  }
```

Aufgabe 21

Ergänzen Sie ihre Matrix-Klasse um eine Matrix-Matrix-Multiplikation. Implementieren Sie diese Multiplikation nicht als Methode, sondern als externe Funktion:

```
Matrix mult(Matrix &A, Matrix &B)
   {
      /* hier ergaenzen */
   }
```

Aufgabe 22

Die Population der Schwäne vom Baldeneysee besteht aus 300 Tieren. Sie sollen zunächst gleichmäßig verteilt sein auf die drei Entwicklungsstufen Küken (x_1), Jungtiere (x_2) und erwachsene Schwäne (x_3), d.h. $x = (100, 100, 100)^T$. Angenommen, jedes Jahr kommen pro Erwachsenem durchschnittlich $a \in \mathbf{R}$ Kücken auf die Welt, und jedes Jahr überlebt der Anteil $d \in [0, 1]$ der Erwachsenen. Der Anteil der Küken, die überleben und zu Jungtieren heranwachsen, ist $b \in [0, 1]$, und der Anteil der Jungtiere, die zu Erwachsenen herangewachsen sind, ist $c \in [0, 1]$.

Dann entsteht aus der Population $x^{(0)} = (100, 100, 100)^T$ die Population $x^{(1)}$ durch Multiplikation mit der Matrix

$$\begin{pmatrix} 0 & 0 & a \\ b & 0 & 0 \\ 0 & c & d \end{pmatrix}$$

Nehmen Sie zunächst an, dass $a = 0.5, b = 0.5, c = 0.5, d = 0.875$. Schreiben Sie ein Programm, dass die Populationen $x^{(n)}$ berechnet und ausgibt.

Verwenden Sie, wenn möglich, die bereits programmierte Matrix-Matrix-Multiplikation (ein Vektor darf dabei als $n \times 1$-Matrix angesehen werden).

Was beobachten Sie für längere Zeiträume? Setzen Sie $d = 0.8$. Wie ist nun das Verhalten für lange Zeiträume?