Masterstudiengang Technische Berechnung und Simulation

Programmierung von CAx-Systemen – Teil 1

Name	Vorname	Matrikelnummer

Aufgabe 1	Aufgabe 2	Aufgabe 3	Summe	

Aufgabensteller: **Dr. Reichl, Dr. Küpper**

Hilfsmittel: **Taschenrechner nicht zugelassen,**
PC/Notebook nicht zugelassen,
sonstige eigene Hilfsmittel sind erlaubt,
Bearbeitung mit Bleistift ist erlaubt.

Viel Erfolg!!!

Aufgabe 1: (ca. 18 Punkte)

Ein C++-Programm soll die folgenden Aufgaben erfüllen:

- Eine Reihe von (positiven) Messwerten wird von der Tastatur eingelesen. Gibt der Anwender einen negativen Messwert oder null ein, wird die Eingabe beendet.
- Von den eingegebenen Messwerten werden die gleitenden Mittelwerte ermittelt. Es wird dazu der arithmetische **Mittelwert von jeweils drei benachbarten Messwerten** berechnet.
- Die berechneten Mittelwerte werden auf dem Bildschirm ausgegeben.

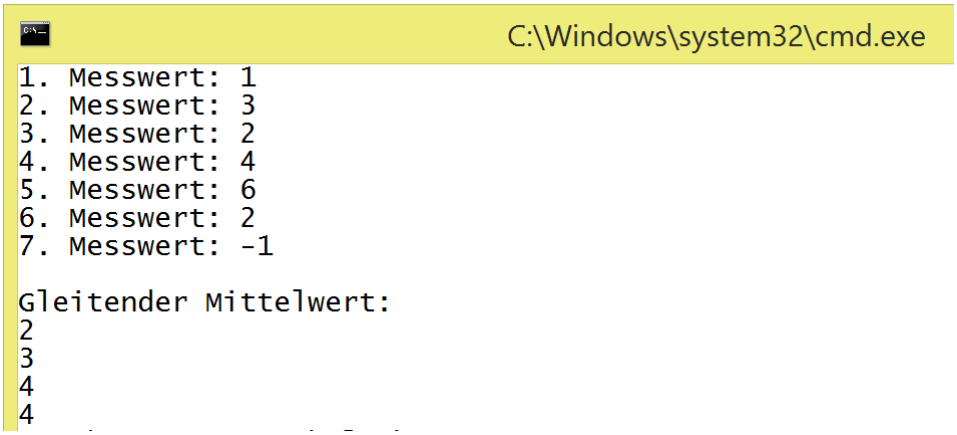
Ergänzen Sie im abgebildeten C++-Quelltext die Funktionen **eingabe()** und **gleitender_mittelwert()**.

Beispiel zur Berechnung der gleitenden Mittelwerte:

Messwerte: 1 3 2 4 6 2 1 3 2 1

Gleitende Mittelwerte: 2 3 4 4 3 2 2 2

Die Anzahl der Mittelwerte ist also immer um 2 kleiner als die Anzahl der Messwerte.




```
// ------------------------------------------------------------------------
// C++-Programm zur Berechnung von gleitenden Mittelwerten
// ------------------------------------------------------------------------

#include <vector>
#include <iostream>

using namespace std;
typedef vector<double> d_vec;

// ------------------------------------------------------------------------
// Die berechneten gleitenden Mittelwerte werden ausgegeben.
// ------------------------------------------------------------------------
void ausgabe(d_vec mittel_vec)
{
    cout << endl << "Gleitender Mittelwert:" << endl;
    for(auto x : mittel_vec)
        cout << x << endl;
}
```

```
// ------------------------------------------------------------------------
// Eine Reihe von Messwerten wird einlesen. Wenn der Anwender null oder einen
// negativen Messwert eingibt, endet die Eingabe. Die Anzahl der Messwerte ist
// nur durch den verfügbaren Hauptspeicher begrenzt. Die Messwerte werden als
// vector<double> zurückgegeben.
// ------------------------------------------------------------------------
d_vec eingabe()
{









}

// ------------------------------------------------------------------------
// Es werden gleitende Mittelwerte aus jeweils drei benachbarten Messwerten
// berechnet und als vector<double> zurückgegeben. Wenn es weniger als drei
// Messwerte gibt, können keine Mittelwerte berechnet werden: In diesem Fall
// wird ein leerer Vektor (also ein Vektor mit null Elementen) zurückgegeben.
// ------------------------------------------------------------------------
d_vec gleitender_mittelwert(d_vec messw)
{









}

// ------------------------------------------------------------------------
// Hauptprogramm
// ------------------------------------------------------------------------
int main(void)
{
    d_vec messwt_vec = eingabe();
    d_vec mittel_vec = gleitender_mittelwert(messwt_vec);
    ausgabe(mittel_vec);
    return 0;
}
```

Aufgabe 2: (ca. 20 Punkte)

Es soll eine neue Klasse zur Arbeit mit reellen 2x2-Matrizen implementiert werden.

Ergänzen Sie im abgebildeten C++-Quelltext den **Konstruktor**, die Methoden **Set()**, **Symm()** und **Max()** sowie die beiden **Operatoren** **+ (Addition)** und **<< (Ausgabe)**.

```
// ------------------------------------------------------------------------
// C++-Programm zur Arbeit mit reellen 2x2-Matrizen
// ------------------------------------------------------------------------

#include <iostream>
using namespace std;

// ------------------------------------------------------------------------
// Definition der Klasse "Matrix2D"
// ------------------------------------------------------------------------
class Matrix2D
{
private:
    // Die einzelnen Elemente der Matrix.
    double a11, a12;
    double a21, a22;

public:
    // Konstruktor: Alle Elemente der Matrix werden auf null gesetzt.
    Matrix2D()
    {




    }

    // Einzelne Elemente der Matrix abfragen.
    double A11() { return a11; }
    double A12() { return a12; }
    double A21() { return a21; }
    double A22() { return a22; }

    // Alle Elemente der Matrix werden auf einen neuen Wert gesetzt.
    void Set(double new11, double new12, double new21, double new22)
    {




    }

    // Ist die Matrix symmetrisch (Rückgabe = 1) oder nicht (Rückgabe = 0)?
    int Symm()
    {




    }
```

```
    // Es wird das größte Element der Matrix ermittelt und zurückgegeben.
    double Max()
    {









    }

    // Zwei Matrizen werden addiert.
    Matrix2D operator+ (Matrix2D m2)
    {









    }
};

// Eine Matrix wird auf einem Stream ausgegeben. Zum Beispiel: cout << mat;
// Die Ausgabe der Matrix erfolgt in dem folgenden Format: [ 1, 2; 3, 4 ]
// (Die einzelnen Elemente werden also durch Kommas bzw. Semikolon unterteilt.
// Am Anfang und am Ende der Matrix wird zudem eine eckige Klammer ausgegeben.)
ostream& operator<< (ostream& strm, Matrix2D mat)
{









}

// ------------------------------------------------------------------------
// Hauptprogramm: Zwei Matrizen addieren, prüfen ob Summe symmetrisch ist,
// maximales Element der Summe ermitteln und auf dem Bildschirm ausgeben.
// ------------------------------------------------------------------------
int main(void)
{
    Matrix2D m1, m2;
    m1.Set(1, 2, 3, 4); m2.Set(4, 3, 0, 1);

    Matrix2D sum = m1 + m2;
    cout << "Summe: " << sum  << endl << "Maximum: " << sum.Max() << endl;

    if(sum.Symm() == 0) cout << "Summe ist nicht symmetrisch." << endl;
    if(sum.Symm() == 1) cout << "Summe ist symmetrisch."       << endl;
    return 0;
}
```

Aufgabe 3: (ca. 12 Punkte)

In einer grafischen, mit Qt erstellten Dialog-Applikation kann eine (Integer-)Zahl eingegeben werden. Nach Betätigung der Schaltfläche „Excel…“ wird eine COM-Verbindung zu Microsoft Excel aufgebaut und es werden einige Werte in das Excel-Tabellenblatt eingetragen.

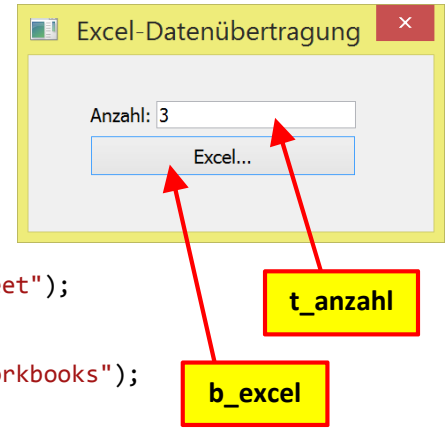
```
void Dialog::on_b_excel_clicked()
{
    int anzahl = ui->t_anzahl->text().toInt();
    if(anzahl < 1) return;  // Eingabefehler

    // Das Attribut "excel" ist vom Typ "QAxWidget",
    // es ist in der Datei dialog.h definiert.
    excel.setControl("Excel.Application");
    excel.setProperty("Visible", true);

    QAxObject *active = excel.querySubObject("ActiveSheet");
    if(!active)
    {
        QAxObject *workbooks = excel.querySubObject("Workbooks");
        workbooks->dynamicCall("Add(void)");
        active = excel.querySubObject("ActiveSheet");
    }

    QAxObject *cells = active->querySubObject("Cells(int,int)", 1, 1);
    cells->setProperty("Value", "Multiplikationstabelle");

    for(int zeile = 1; zeile <= anzahl; ++zeile)
    {
        for(int spalte = 1; spalte <= anzahl; ++spalte)
        {
            cells = active->querySubObject("Cells(int,int)", zeile + 2, spalte);
            cells->setProperty("Value", zeile * spalte);
        }
    }
}
```



Der abgebildete Quelltext-Ausschnitt zeigt den Slot, der bei Betätigung der Schaltfläche „Excel…“ aufgerufen wird: Welche Werte werden in das Excel-Tabellenblatt eingetragen, wenn der Anwender in der grafischen Benutzeroberfläche **die Zahl 3 eingegeben** und anschließend die Schaltfläche „Excel…“ betätigt hat?