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, objektorientierte Programmierung: (ca. 18 Punkte)

Es soll eine neue Klasse **Widerstand** zum Rechnen mit ohmschen Widerständen programmiert werden. Der aktuelle Widerstandswert ist im privaten **Attribut r** gespeichert. Ergänzen Sie die fehlenden Methoden und Funktionen!

Hinweise: - Für die Reihenschaltung von Widerständen gilt: $R_{ges} = R_a + R_b$
- Für die Parallelschaltung von Widerständen gilt: $1/R_{ges} = 1/R_a + 1/R_b$
- Das ohmsche Gesetz lautet: I = U/R mit I = Strom, U = Spannung, R = Widerstand

```
// Programm zum Rechnen mit ohmschen Widerständen
#include <iostream>
using namespace std;

// Definition der neuen Klasse "Widerstand"
class Widerstand
{
private:
    // aktueller Widerstandswert (in Ohm)
    double r;

public:
    // Default-Konstruktor: Widerstand auf 1000 Ohm setzen
    Widerstand()
    {




    }

    // Alternativer Konstruktor: Widerstand auf angegebenen Wert setzen
    Widerstand(double r_neu)
    {




    }

    // Der Widerstand wird auf einen neuen Wert gesetzt
    void SetR(double r_neu)
    {




    }

    // Der aktuelle Widerstandswert wird abgefragt
    double GetR()
    {




    }

    // Ohmsches Gesetz (a): Für die als Parameter übergebene Spannung
    // wird der Strom berechnet, der durch den Widerstand fließt
    double BerechneStrom(double u)
    {




    }
```

```
    // Ohmsches Gesetz (b): Für den als Parameter übergebenen Strom
    // wird die Spannung berechnet, die am Widerstand abfällt
    double BerechneSpannung(double i)
    {




    }
};

// Gesamtwiderstand der Reihenschaltung von a und b berechnen und zurückgeben
Widerstand operator+ (Widerstand a, Widerstand b)
{









}

// Gesamtwiderstand der Parallelschaltung von a und b berechnen und zurückgeben
Widerstand operator* (Widerstand a, Widerstand b)
{









}

// Widerstandswert auf Ausgabestream "strm" schreiben. Die Ausgabe
// soll in der Form "10.5 Ohm" erfolgen, als Zahlenwert mit Einheit.
ostream& operator<< (ostream& strm, Widerstand a)
{









}

// ------------------------------------------------------------------------
// Hauptprogramm
// ------------------------------------------------------------------------
int main()
{
    Widerstand r1(100), r2(1000), r_par, r_ser;
    r_par = r1 * r2;  // Berechnung der Parallelschaltung
    r_ser = r1 + r2;  // Berechnung der Reihenschaltung
    cout << "Reihenschaltung:   " << r_ser << endl;
    cout << "Parallelschaltung: " << r_par << endl;

    auto i = r1.BerechneStrom(10.0);
    auto u = r1.BerechneSpannung(1e-3);
    cout << "Strom durch r1 bei 10V = " << i << "A" << endl;
    cout << "Spannung an r1 bei 1mA = " << u << "V" << endl;
    return 0;
}
```

Aufgabe 2, C++-Standardbibliothek: (ca. 22 Punkte)

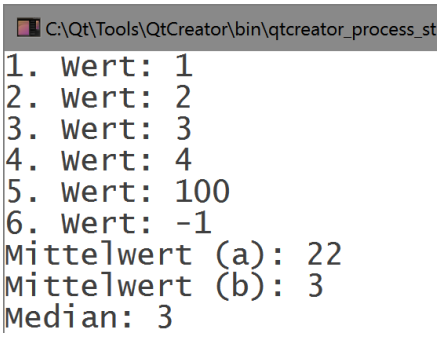
Ein C++-Programm soll die folgenden Aufgaben erfüllen:

- 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 Messwerten werden der arithmetische Mittelwert (Funktion **mean1**) und der Median (Funktion **median**) berechnet.
- Mit der Funktion **mean2** wird nochmals der arithmetische Mittelwert berechnet, allerdings ohne (!) Berücksichtigung des größten und des kleinsten Messwerts.

Hinweise:

- Der Median einer Liste von Zahlenwerten ist der Wert, der an der mittleren (zentralen) Stelle steht, wenn man die Werte der Größe nach sortiert. Beispielsweise ist für die Werte 4, 1, 37, 2, 1 die Zahl 2 der Median, nämlich die mittlere Zahl in 1, 1, 2, 4, 37.

  Bei einer geraden Anzahl von Werten ist der Median das arithmetische Mittel der beiden mittleren Zahlen. (Quelle: Wikipedia/Median)
- Tipp: Durch Aufruf von **sort(werte.begin(), werte.end());** können die Werte im Vektor sortiert werden.




```
// C++-Programm zur Berechnung von Median und Mittelwert
#include <iostream>   // std::cout
#include <vector>     // std::vector
#include <algorithm>  // std::sort
using namespace std;

// Typdefinition: Vektor mit double-Elementen
typedef vector<double> d_vec;

// Der Anwender gibt beliebig viele Werte ein. Die Eingabe endet,
// wenn 0 oder ein negativer Wert eingegeben wird. Ein Vektor mit
// allen eingegebenen Werten wird als Rückgabewert zurückgegeben.
d_vec eingabe(void)
{









}
```

```
// Mittelwert von allen Werten im Vektor berechnen und zurückgeben.
// Bei einem leeren Vektor wird 0 zurückgegeben!
double mean1(d_vec werte)
{









}

// Mittelwert der Werte im Vektor berechnen, der größte und der kleinste
// Wert werden nicht berücksichtigt! Wenn der Vektor aus weniger als drei
// Elementen besteht, ist der Rückgabewert identisch zur Funktion mean1().
double mean2(d_vec werte)
{
```



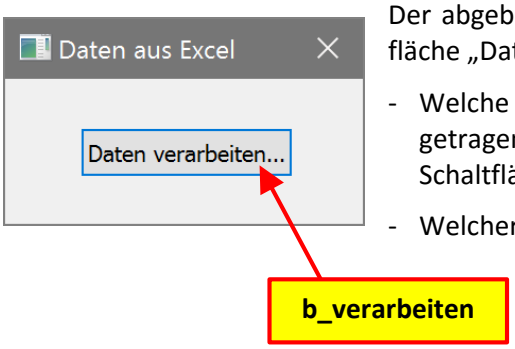
```
}

// Median der Werte im Vektor berechnen und zurückgeben.
// Achtung: Die Werte im Vektor sind in der Regel nicht sortiert!
double median(d_vec werte)
{
```



```
}

// Hauptprogramm
int main()
{
    auto test = eingabe();
    cout << "Mittelwert (a): " << mean1 (test) << endl;
    cout << "Mittelwert (b): " << mean2 (test) << endl;
    cout << "Median: "         << median(test) << endl;
    return 0;
}
```

Aufgabe 3, Benutzeroberflächen, COM-Schnittstellen: (ca. 10 Punkte)

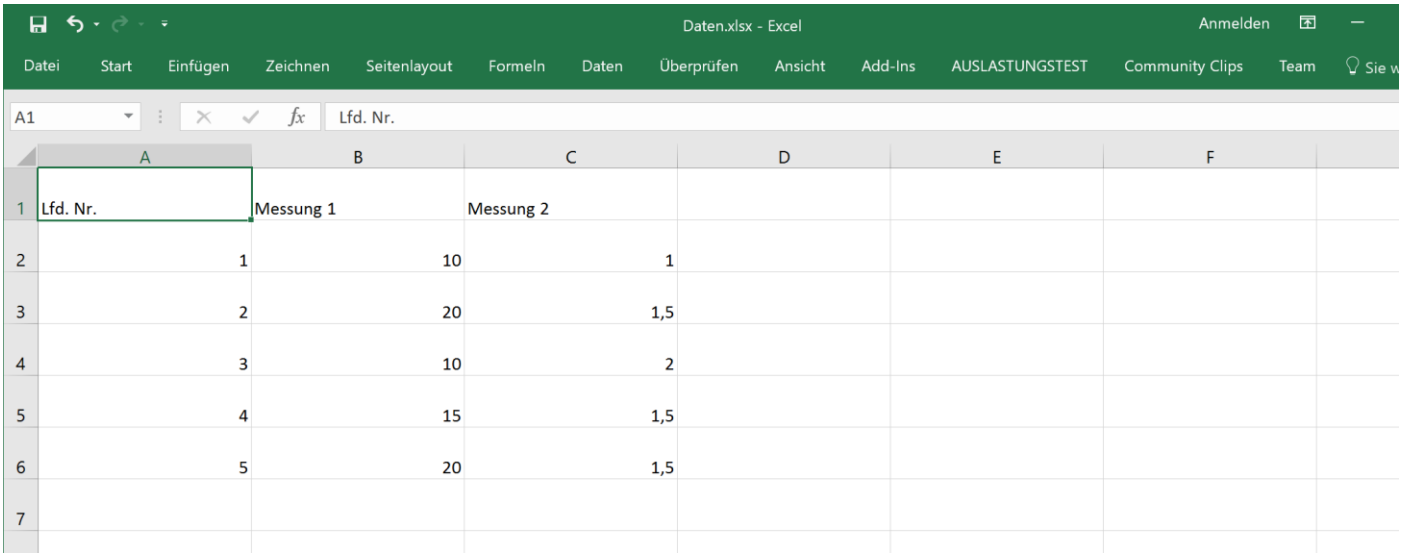
Das abgebildete C++-Programm öffnet eine COM-Schnittstelle zu Microsoft Excel, liest einige Werte aus der in Excel geöffneten Arbeitsmappe, führt einfache Berechnungen durch und überträgt die Ergebnisse wieder nach Excel.



Der abgebildete Quelltext-Ausschnitt zeigt den Slot, der bei Betätigung der Schaltfläche „Daten verarbeiten…“ aufgerufen wird:

- Welche Werte werden in das Tabellenblatt auf der nebenstehenden Seite eingetragen, nachdem der Anwender in der grafischen Benutzeroberfläche die Schaltfläche betätigt hat?
- Welcher Text wird anschließend in einem Meldungsfenster angezeigt?

```
void Dialog::on_b_verarbeiten_clicked()
{
    // Das Attribut "excel" ist vom Typ "QAxWidget"
    excel.setControl("Excel.Application");
    excel.setProperty("Visible", true);

    auto *workbooks = excel.querySubObject("Workbooks");
    workbooks->dynamicCall("Open(QString)", "C:/Temp/Daten.xlsx");

    auto *active = excel.querySubObject("ActiveSheet");
    if(!active) return;  // Datei nicht gefunden...?

    double sum1 = 0, sum2 = 0;
    int anzahl = 0;

    auto cell1 = active->querySubObject("Cells(int,int)", 2, 2);
    auto mess1 = cell1->property("Value");
    auto cell2 = active->querySubObject("Cells(int,int)", 2, 3);
    auto mess2 = cell2->property("Value");

    while(!mess1.isNull() && !mess2.isNull())
    {
        ++anzahl;
        sum1 += mess1.toDouble();
        sum2 += mess2.toDouble();

        cell1 = active->querySubObject("Cells(int,int)", 2 + anzahl, 2);
        mess1 = cell1->property("Value");
        cell2 = active->querySubObject("Cells(int,int)", 2 + anzahl, 3);
        mess2 = cell2->property("Value");
    }

    cell1 = active->querySubObject("Cells(int,int)", 1, 5);
    cell1->setProperty("Value", "Ergebnis 1:");
    cell1 = active->querySubObject("Cells(int,int)", 1, 6);
    cell1->setProperty("Value", sum1 / anzahl);

    cell2 = active->querySubObject("Cells(int,int)", 2, 5);
    cell2->setProperty("Value", "Ergebnis 2:");
    cell2 = active->querySubObject("Cells(int,int)", 2, 6);
    cell2->setProperty("Value", sum2 / anzahl);

    ostringstream strm;
    strm << "Es wurden " << anzahl << " Zeilen verarbeitet.";

    QMessageBox box;
    box.setText(strm.str().c_str());
    box.exec();
}
```


Daten.xlsx - Excel
Anmelden
Datei
Start
Einfügen
Zeichnen
Seitenlayout
Formeln
Daten
Überprüfen
Ansicht
Add-Ins
AUSLASTUNGSTEST
Community Clips
Team
A1
Lfd. Nr.
Lfd. Nr.
Messung 1
Messung 2
1 10 1
2 20 1,5
3 10 2
4 15 1,5
5 20 1,5


A3
OK

(Platz für Notizen und Nebenrechnungen)