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: Programmierung mit der C++-Standardbibliothek (ca. 19 Punkte)

Vervollständigen Sie das vorbereitete C++-Programm zur Berechnung von Polynomen. Die Koeffizienten eines Polynoms werden im C++-Programm als double-Vektor abgespeichert, zum Beispiel:

- Das Polynom $p(x) = 1x^3 - 2x^2 + 3x - 4$ wird abgespeichert als Vektor mit 4 Elementen: 1, -2, 3, -4
- Das Polynom $p(x) = 2x^2 - 4x + 6$ wird abgespeichert als Vektor mit 3 Elementen: 2, -4, 6

Im unten abgedruckten Quelltext der Funktion main() finden Sie ein weiteres Beispiel.

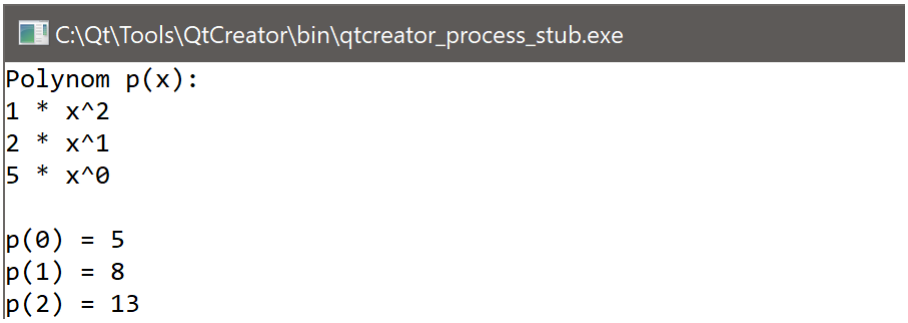
1.1. Programmieren Sie die Funktion polyprint() zur Ausgabe eines Polynoms auf einem C++-Stream. Die Ausgabe soll so aussehen, wie im Bildschirmfoto gezeigt (Glieder des Polynoms untereinander ausgeben).

1.2. Programmieren Sie die Funktion polyval1(), die das Polynom p(x) an einer einzelnen x-Position ausrechnet und das Ergebnis als Rückgabewert zurückgibt.

1.3. Programmieren Sie die Funktion polyval2(), die das Polynom p(x) an allen x-Positionen ausrechnet, die im Vektor xx übergeben werden. Es wird ein Vektor mit allen Ergebnissen zurückgegeben. Tipp: Rufen Sie an einer geeigneten Stelle innerhalb von polyval2() die bereits existierende Funktion polyval1() auf…

1.4. Erweitern bzw. verändern Sie das Hauptprogramm main() so, dass die Ausgabe des Programms nicht mehr auf dem Bildschirm, sondern in einer Textdatei "c:/temp/out.txt" erfolgt.

Das Bildschirmfoto zeigt die Ausgabe des unten abgedruckten Hauptprogramms:

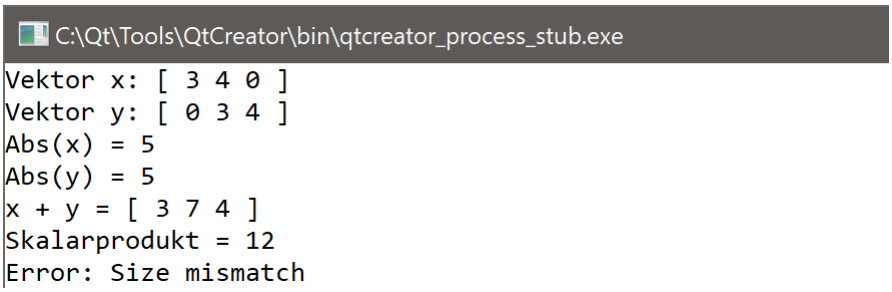



```
// ------------------------------------------------------------------------
// C++-Programm zur Berechnung von Polynomen
// ------------------------------------------------------------------------
#include <iostream>
#include <vector>
#include <fstream>
#include <math.h>  // ggf. für pow()-Funktion?

using namespace std;
typedef vector<double> d_vec;

// Deklarationen
void polyprint(ostream& strm, d_vec p);
double polyval1(d_vec p, double x);
d_vec polyval2(d_vec p, d_vec xx);

// ------------------------------------------------------------------------
// Hauptprogramm
// ------------------------------------------------------------------------
int main(void)
{
    // Aufgabe 1.4:
    // Erweitern bzw. verändern Sie das Hauptprogramm main() so,
    // dass die Ausgabe des Programms nicht mehr auf dem Bildschirm,
    // sondern in einer Textdatei "c:/temp/out.txt" erfolgt.
```

```
    // Polynom erzeugen: p(x) = 1*x^2 + 2*x + 5
    d_vec p;
    p.push_back(1.0); p.push_back(2.0); p.push_back(5.0);

    // Polynom ausgeben, siehe Bildschirmfoto!
    cout << "Polynom p(x):" << endl;
    polyprint(cout, p);

    // Polynom p(x) für x=0, x=1 und für x=2 ausrechnen und ausgeben
    d_vec xx;
    xx.push_back(0.0); xx.push_back(1.0); xx.push_back(2.0);
    auto yy = polyval2(p, xx);  // mehrere x-Positionen werden übergeben!
    for(int i = 0; i < (int)p.size(); ++i)
        cout << "p(" << xx[i] << ") = " << yy[i] << endl;
}

// ----------------------------------------------------------------------------
// Glieder des Polynoms p(x) untereinander ausgeben, Aufgabe 1.1
// ----------------------------------------------------------------------------
void polyprint(ostream& strm, d_vec p)
{









}

// ----------------------------------------------------------------------------
// Polynom p(x) an einer einzelnen x-Position ausrechnen, Aufgabe 1.2
// ----------------------------------------------------------------------------
double polyval1(d_vec p, double x)
{









}

// ----------------------------------------------------------------------------
// Polynom p(x) an mehreren x-Positionen ausrechnen, Aufgabe 1.3
// ----------------------------------------------------------------------------
d_vec polyval2(d_vec p, d_vec xx)
{









}
```

Aufgabe 2: Objektorientierte Programmierung (ca. 26 Punkte)

Programmieren Sie eine Klasse Mathvect zum Rechnen mit Vektoren: Die einzelnen Vektor-Elemente sind in der internen Variablen (bzw. dem Attribut) „data“ abgespeichert. Auf die einzelnen Vektor-Elemente kann „von außen“ über Get()- bzw. Set()-Methoden zugegriffen werden. Zwei Vektoren können addiert werden, auch die Berechnung des Skalarprodukts ist möglich. Die Methode ToString() gibt den Inhalt eines Vektors als Zeichenkette (std::string) zurück. Das Bildschirmfoto zeigt die Ausgabe des unten abgedruckten Hauptprogramms.



2.1. Programmieren Sie den Konstruktor der Klasse Mathvect, der einen neuen Vektor mit „dim“ Vektor-Elementen erzeugt. Alle Vektor-Elemente sollen im Konstruktor auf null gesetzt werden.

2.2. Programmieren Sie die Methode Dim(), welche die Anzahl der Vektor-Elemente zurückgibt.

2.3. Programmieren Sie die Methode Get(), welche das Vektor-Element an Position „idx“ zurückgibt (Achtung: nullbasierte Zählung verwenden, das erste Vektor-Element hat den Index 0).

Falls beim Aufruf dieser Methode ein ungültiger Index übergeben wird (Index zu groß oder Index kleiner als 0), dann soll die Fehlermeldung „Index invalid“ ausgegeben und das Programm abgebrochen werden. Verwenden Sie dazu die vorbereitete Funktion error().

2.4. Programmieren Sie die Methode Set(), welche das Vektor-Element an Position „idx“ auf einen neuen Wert setzt (wieder nullbasierte Zählung!). Auch bei dieser Methode soll das Programm mit der Fehlermeldung „Index invalid“ abgebrochen werden, wenn ein ungültiger Index übergeben wird.

2.5. Programmieren Sie die Methode Abs(), welche den Betrag (auch: „Länge“, euklidische Norm, 2-Norm) des Vektors berechnet und zurückgibt.

2.6. Programmieren Sie einen operator+ zur Addition von zwei Vektoren. Wenn die Element-Anzahl der zu addierenden Vektoren unterschiedlich ist, dann soll das Programm mit der Fehlermeldung „Size mismatch“ abgebrochen werden. Verwenden Sie zur Ausgabe der Fehlermeldung die vorbereitete Funktion error().

2.7. Programmieren Sie die Funktion dot() zur Berechnung des Skalarprodukts von zwei Vektoren. Auch hier soll das Programm mit der Fehlermeldung „Size mismatch“ abgebrochen werden, wenn die Element-Anzahl der beiden Vektoren a und b unterschiedlich ist.

2.8. Programmieren Sie die Methode ToString(), welche den Inhalt des Vektors als Zeichenkette (std::string) zurückgibt. Beachten Sie die Beispiele im unten abgedruckten Hauptprogramm main() sowie das Bildschirmfoto. Die Zeichenkette soll so aussehen, wie im Bildschirmfoto dargestellt. Tipp: std::stringstream verwenden…

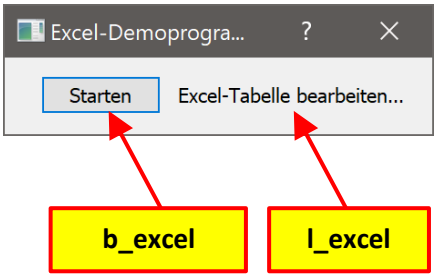
```
#include <math.h>     // sqrt()
#include <stdlib.h>   // exit()
#include <iostream>
#include <vector>
#include <string>
#include <sstream>    // std::stringstream

using namespace std;
typedef vector<double> d_vec;

// ---------------------------------------------------------------------------
// Fehlermeldung auf dem Bildschirm ausgeben, Programm beenden
// ---------------------------------------------------------------------------
void error(string meldung)
{
    cout << "Error: " << meldung << endl;
    exit(EXIT_FAILURE);  // Programm beenden
}
```

```
// ---------------------------------------------------------------------------
// Definition der Klasse Mathvect
// ---------------------------------------------------------------------------
class Mathvect
{
private:
    // Die einzelnen Vektor-Elemente sind hier gespeichert!
    d_vec data;

public:
    // Vektor mit "dim" Elementen erzeugen, Elemente auf 0 setzen, Aufgabe 2.1
    Mathvect(int dim)
    {









    }

    // Anzahl der Vektor-Elemente zurückgeben, Aufgabe 2.2
    int Dim(void)
    {



    }

    // Element an Position "idx" abfragen (nullbasierter Index), Aufgabe 2.3
    double Get(int idx)
    {









    }

    // Element an Position "idx" auf einen neuen Wert setzen, Aufgabe 2.4
    void Set(int idx, double value)
    {









    }

    // Betrag (Länge, euklidische Norm, 2-Norm) des Vektors berechnen, Aufgabe 2.5
    double Abs(void)
    {









    }
```

```
    // Zwei gleich große Vektoren addieren, Aufgabe 2.6
    Mathvect operator+ (Mathvect v)
    {









    }

    // Inhalt des Vektors als Zeichenkette (std::string) zurückgeben, Aufgabe 2.8
    string ToString(void)
    {









    }
};

// Skalarprodukt von zwei gleich großen Vektoren berechnen, Aufgabe 2.7
double dot(Mathvect a, Mathvect b)
{









}

// ----------------------------------------------------------------------------
// Hauptprogramm
// ----------------------------------------------------------------------------
int main(void)
{
    Mathvect x(3), y(3), z(3), test(10);
    x.Set(0, 3.0); x.Set(1, 4.0); x.Set(2, 0.0);
    y.Set(0, 0.0); y.Set(1, 3.0); y.Set(2, 4.0);

    cout << "Vektor x: " << x.ToString() << endl;
    cout << "Vektor y: " << y.ToString() << endl;
    cout << "Abs(x) = " << x.Abs() << endl;
    cout << "Abs(y) = " << y.Abs() << endl;

    z = x + y;
    cout << "x + y = " << z.ToString() << endl;

    double s = dot(x, y);
    cout << "Skalarprodukt = " << s << endl;

    z = x + test;  // Fehler ("Size mismatch")...
}
```

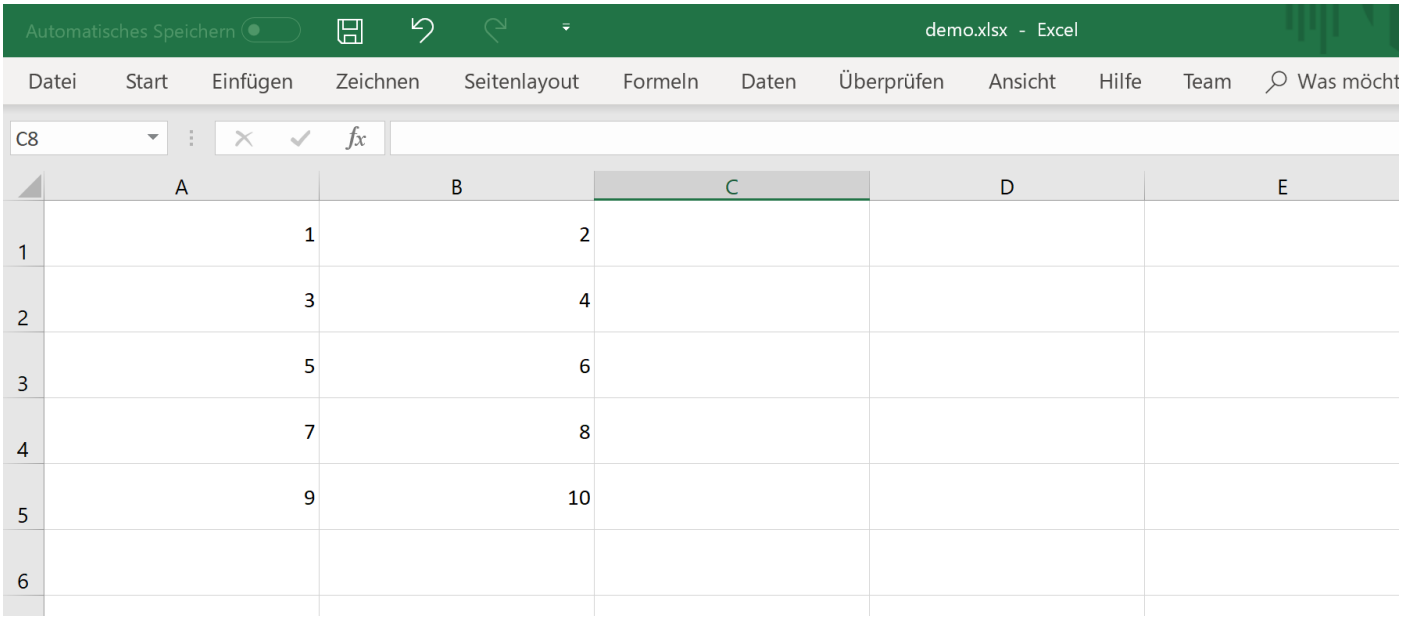
Aufgabe 3: Grafische Benutzeroberflächen, Softwareschnittstellen (ca. 5 Punkte)

Die abgebildete Benutzeroberfläche wurde mit QtWidgets erstellt. Das Programm öffnet eine Excel-Tabelle, die sich bereits auf der Festplatte befindet, und bearbeitet die darin gespeicherten Daten.



Vor dem Start des Programms befinden sich in der Excel-Tabelle die unten auf dieser Seite gezeigten Zahlen. Wie sieht die Excel-Tabelle nach Ausführung der Methode „on_b_excel_clicked()“ aus?

```
void Dialog::on_b_excel_clicked()
{
    QVariant a, b;
    QAxObject excel, *workbooks, *active, *cell;

    // Excel-Tabelle öffnen...
    excel.setControl("Excel.Application");
    excel.setProperty("Visible", true);

    workbooks = excel.querySubObject("Workbooks");
    workbooks->dynamicCall("Open(QString)", "c:/temp/demo.xlsx");
    active = excel.querySubObject("ActiveSheet");
    if(!active) return;  // ...existiert nicht?!

    // Werte bearbeiten...
    int zeile = 1;
    cell = active->querySubObject("Cells(int,int)", zeile, 1);
    a = cell->property("Value");
    cell = active->querySubObject("Cells(int,int)", zeile, 2);
    b = cell->property("Value");

    while(!a.isNull() && !b.isNull())
    {
        double prod = a.toDouble() * b.toDouble();
        cell = active->querySubObject("Cells(int,int)", zeile, 3);
        cell->setProperty("Value", prod);

        ++zeile;
        cell = active->querySubObject("Cells(int,int)", zeile, 1);
        a = cell->property("Value");
        cell = active->querySubObject("Cells(int,int)", zeile, 2);
        b = cell->property("Value");
    }
}
```



(Platz für Notizen und Nebenrechnungen)