Hochschule München University of Applied Sciences

Sommersemester 2021

Programmierung von CAx-Systemen – Teil 1

Schriftliche Fernprüfung mit Videoaufsicht

Prüfer: Küpper, Reichl

Bearbeitungszeit für beide Teile: 60 Minuten

Hilfsmittel:

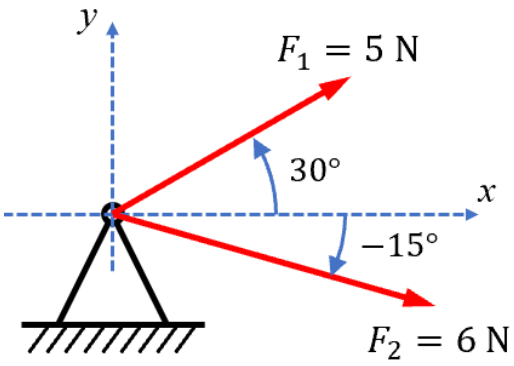
- **Taschenrechner und elektronische Hilfsmittel sind nicht zugelassen.**
- **Alle schriftlichen Unterlagen sind erlaubt.**

Schreiben Sie Ihren Namen, Vornamen und auch die Studiengruppe auf alle Lösungsblätter. Es werden nur handschriftliche Lösungen auf leeren, weißen DIN-A4-Blättern akzeptiert.

****** Viel Erfolg! ******

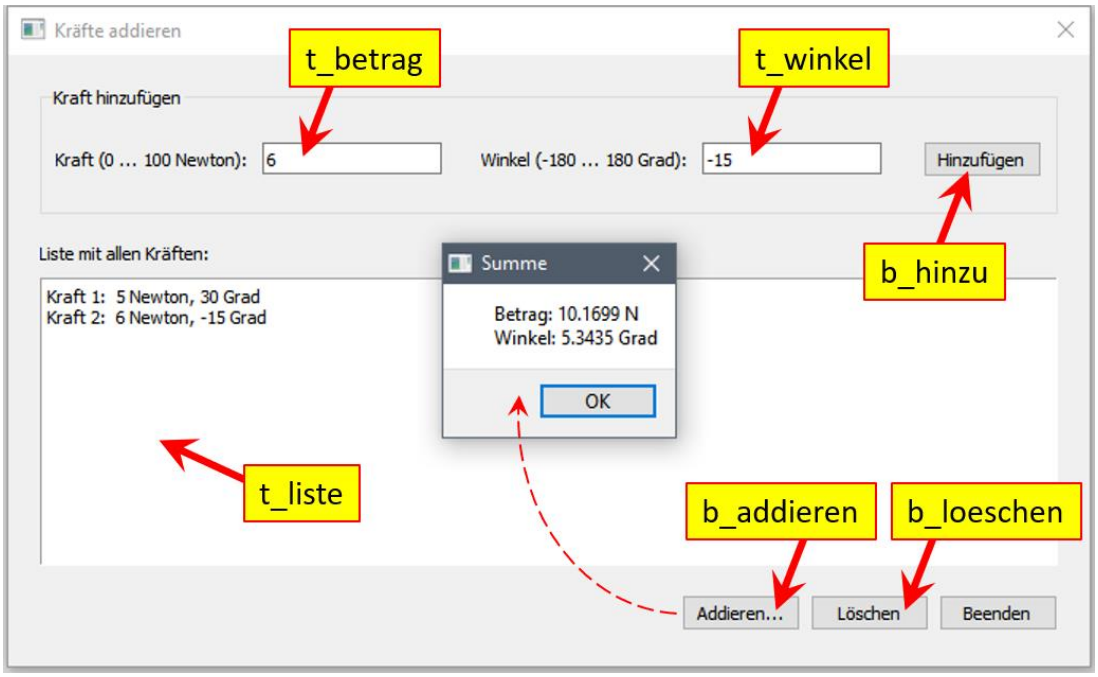
Aufgabe 1: C++-Standardbibliothek, GUI-Programmierung (ca. 21 Punkte)

Auf ein Festlager wirken mehrere Kräfte F_1, F_2, F_3 usw. wie im Bild rechts dargestellt. (Im Bild sind nur zwei Kräfte F_1 und F_1 abgebildet.) Die Beträge der einzelnen Kräfte liegen jeweils im Bereich von 0 … 100 N, die Winkel liegen im Bereich von $-180°$ … $+180°$.



Es soll eine QtWidgets-Dialogapplikation erstellt werden, mit der die einzelnen Kräfte addiert werden können:

- Der Anwender gibt den Betrag und den Winkel einer Kraft in den QLineEdit-Feldern t_betrag und t_winkel ein. Mit der Schaltfläche b_hinzu wird diese neue Kraft hinzugefügt.
- Die Liste aller eingegebenen Kräfte wird im QTextEdit-Feld t_liste angezeigt.
- Durch Druck auf die Schaltfläche b_addieren wird die Gesamtkraft F_{ges} ausgerechnet. Der Betrag und der Winkel von F_{ges} werden in einer QMessageBox ausgegeben.



Die Klasse Dialog ist wie folgt deklariert. Beachten Sie, dass in den beiden Vektoren v_betrag und v_winkel die Beträge und Winkel von allen eingegebenen Kräften abgespeichert sind:

```
class Dialog : public QDialog
{
    Q_OBJECT

public:
    Dialog(QWidget *parent = nullptr);
    ~Dialog();

private slots:
    void on_b_hinzu_clicked();
    void on_b_addieren_clicked();
    void on_b_loeschen_clicked();
    void on_b_beenden_clicked();

private:
    // Zwei Vektoren mit allen Kräften und Winkeln:
    std::vector<double> v_betrag, v_winkel;
    Ui::Dialog *ui;
};
```

In der Datei dialog.h sind alle notwendigen Header-Dateien eingebunden:

```
#include <QDialog>
#include <QMessageBox>
#include <vector>
#include <string>
#include <sstream>
#include <math.h>
using namespace std;
```

1.1 Die Methode Dialog::on_b_hinzu_clicked() wird aufgerufen, wenn der Anwender die Schaltfläche b_hinzu drückt.

Wie lauten die Befehle zu den – noch fehlenden – Schritten c, d, und e?

```
void Dialog::on_b_hinzu_clicked()
{
    // a. String-Stream zur Ausgabe in t_liste.
    ostringstream strm;

    // b. Werte in t_betrag und t_winkel ermitteln.
    bool ok1, ok2;
    double betrag = this->ui->t_betrag->text().toDouble(&ok1);
    double winkel = this->ui->t_winkel->text().toDouble(&ok2);

    // c. Es wird geprüft: Ist der Betrag im Bereich -100 ... 100 und
    // ist der Winkel im Bereich -180 ... 180? Wurden überhaupt zwei
    // gültige Zahlen eingegeben? Falls die Eingabe ungültig ist,
    // wird eine Fehlermeldung in einer QMessageBox ausgegeben.

    // d. Der eingegebene Betrag und der eingegebene Winkel werden als
    // neue Elemente am Ende der Vektoren v_betrag und v_winkel hinzugefügt.

    // e. Alle Beträge und Winkel aus den Vektoren v_betrag und v_winkel
    // werden der Reihe nach durchlaufen und als "lesbarer Text" in den
    // String-Stream geschrieben. Der Text im String-Stream soll so aus-
    // sehen, wie im Bildschirmfoto gezeigt (siehe Feld t_liste).

    // f. String-Stream nach t_liste übertragen.
    ui->t_liste->setText(strm.str().c_str());
}
```

1.2 Die Methode Dialog::on_b_addieren_clicked() wird aufgerufen, wenn der Anwender die Schaltfläche b_addieren drückt. Das Programm berechnet nun die Gesamtkraft F_{ges} und gibt das Ergebnis in einer QMessageBox auf dem Bildschirm aus. (Siehe Bildschirmfoto auf der vorherigen Seite.)

Falls noch gar keine Kräfte eingegeben wurden, wird stattdessen die Meldung „Keine Kräfte eingegeben!“ in einer QMessageBox angezeigt.

Wie lauten die Befehle zu den – noch fehlenden – Schritten a, b und c?

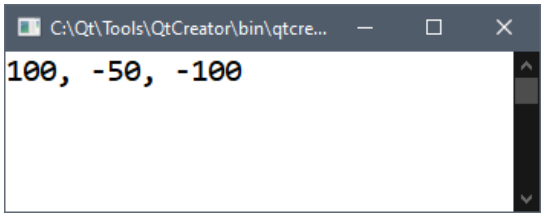
```
void Dialog::on_b_addieren_clicked()

{
    // a. Fehlermeldung ausgeben, falls die Vektoren
    // v_betrag bzw. v_winkel noch leer sind; die Schritte
    // b und c werden in diesem Fall nicht ausgeführt.

    // b. Alle Kräfte in den Vektoren v_betrag und v_winkel
    // durchlaufen und die Gesamtkraft (mit Betrag und Winkel)
    // berechnen.

    // c. Betrag und Winkel der Gesamtkraft in einer QMessageBox
    // auf dem Bildschirm ausgeben (siehe Bildschirmfoto).
}
```

Aufgabe 2: Programmierung von eigenen Klassen (ca. 13 Punkte)

Es soll eine (neue, bisher noch nicht vorhandene) Klasse LimitedDouble programmiert werden. LimitedDouble-Variablen können ganz ähnlich wie double-Variablen verwendet werden. Der Wert einer LimitedDouble-Variablen kann allerdings niemals größer als 100.0 und niemals kleiner als -100.0 werden.

Das Hauptprogramm zeigt einige Beispiele zur Verwendung von LimitedDouble-Variablen. Beachten Sie auch das dazugehörige Bildschirmfoto.

Wie lauten die Befehle zu den Schritten a, b, c, d und e?




```
#include <iostream>
using namespace std;

class LimitedDouble
{
private:
    double value;  // Aktueller Wert

public:
    // Erster Konstruktor: Aktuellen Wert auf null setzen.
    LimitedDouble() { SetValue(0); }

    // Zweiter Konstruktor: Aktuellen Wert auf d setzen. Achtung: Wenn d größer
    // als 100 ist, dann wird der aktuelle Wert auf 100 gesetzt. Wenn d kleiner
    // als -100 ist, wird der aktuelle Wert auf -100 gesetzt.
    LimitedDouble(double d)
    {
        // Die Bereichs-Überprüfung -100 ... 100 wird von SetValue() erledigt.
        SetValue(d);
    }

    // Aktuellen Wert zurückgeben.
    double GetValue()
    {
        // a.      ... TODO ...
    }

    // Der aktuelle Wert wird auf d gesetzt. Achtung: Wenn d größer als 100 ist,
    // dann wird der aktuelle Wert auf 100 gesetzt. Wenn d kleiner als -100 ist,
    // dann wird der aktuelle Wert auf -100 gesetzt.
    void SetValue(double d)
    {
        // b.      ... TODO ...
    }
};

// Programmieren Sie den operator+ zum Addieren von zwei LimitedDouble-Variablen.
// c.      ... TODO ...

// Programieren Sie den operator- zum Subtrahieren von zwei LimitedDouble-Variablen.
// d.      ... TODO ...

// Programmieren Sie den operator<< zur Ausgabe einer LimitedDouble-Variablen.
// e.      ... TODO ...

int main(void)
{
    LimitedDouble x1(75.0), x2(55.5);
    LimitedDouble x3 = x1 + x2;       // Limit +100 erreicht
    LimitedDouble x4 = x3 - x1 - x1;  // Ergebnis = -50
    LimitedDouble x5 = x4 - x1;       // Limit -100 erreicht

    cout << x3 << ", " << x4 << ", " << x5 << endl;  // Ausgabe: 100, -50, -100
    return 0;
}
```