Übung Notendarstellung

Einführung

Unsere Zeichenfläche hat einige Erweiterungen erfahren, die wir heute nutzen wollen um eine Notendarstellung zu erhalten. Insbesondere hat gibt es nun die Möglichkeit Linien und Punkte auf dem Bildschirm darzustellen.

Erster Schritt: Klasse Note

Diese Klasse mag ein wenig überraschen, denn wofür braucht es um eine Note zu speichern eine Klasse? Dafür genügt doch auch einfach ein double! Oder ein float? Es schadet nicht eine Note in einer Klasse zu kapseln, der Datentyp mit dem der Notenwert gespeichert wird ist dann gekapselt. Als Methode kann unsere Notenklasse einen gerundetet Wert ausgeben, was ja nicht so schlecht ist! Wie wäre es mit folgender Klasse?

```
#ifndef NOTE_H
#define NOTE_H

class Note
{
    public:
        // Konstruktor
        Note();
        // Konstruktor mit Parameter
        Note(double note);
        // Kopierkonstruktor
        Note(const Note& vorlage);

        // nachträgliches Setzen der Note
        void setzeNote(double note);

        double holeNote();

        enum Rundung
        {
            Zehntel,
            Viertel,
            Halbe,
            Ganze
        };
        // gibt einen gerundeten Wert zurück
        // gemäss Parameter
        double holeGerundeteNote(Rundung rundung);
    private:
        double m_notenWert;
};

#endif
```

Zweiter Schritt : Klasse Notenliste

Wie in der letzten Übung CDListe schreiben wir hier auch eine Klasse, die viele Noten speichern kann. Am besten du haltest Dich an die Vorgabe. Ersetze einfach die Klasse CD mit der Klasse Note. Hier was ich in einer ersten Version implementieren würde:

```
#ifndef NOTENLISTE_H
#define NOTENLISTE_H

#include <string>
#include "Note.h"

const unsigned long MaxNoten = 50;

class Notenliste
{
    public:
        // Konstruktor
        Notenliste(const std::string& Fach);

        bool hinzufuegen(const Note& note);

        // Diese Funktion berechnet den
        // Durchschnitt aller Noten
        Note berechneDurchschnitt();

    private:
        Note        m_Noten[MaxNoten];
        int         m_AnzahlNoten;
        std::string m_Fach;
};

#endif
```

Mir gefällt hier vor allem die Funktion *berechneDurchschnitt*, denn sie gibt eine Note als Rückgabewert, statt eines schnöden doubles oder so. Der Vorteil ist, dass ich diese Note auch wieder runden kann (mit Note:: holeGerundeteNote(...)).

Dritter Schritt : Ausgabe und Testen

Wie du bemerkt hast, haben wir bis jetzt rein gar nichts auf dem Bildschirm ausgegeben! Weder grafisch noch sonst irgendwie. Zusammen fügen wir jetzt eine nette Ausgabefunktion mit ein wenig Zauberfähigkeit hinzu. Ergänze folgende Funktion in der Klasse Notenliste:

```
void ausgeben(ostream& out);
```

Der Parameter vom Typ ostream entspricht dem cout – Objekt. Aber nicht nur! Wir können nämlich anstelle des cout – Objektes auch ein Datei - Objekt der Funktion *ausgeben* mitgeben. Füge auch die eine solche Funktion *ausgeben* in die Klasse *Note* ein. Hier die eine mögliche Implementation dieser beiden Funktionen.

```
void Note::ausgeben(ostream& out)
{
    out << m_notenWert << endl;
}


void Notenliste::ausgeben(ostream& out)
{
    out << "Noten des Fachs " << m_Fach << endl;
    out << endl;
    out << "Anzahl Noten : " << m_AnzahlNoten << endl;
    out << endl;
    for(int i = 0; i < m_AnzahlNoten; ++i)
    {
        m_Noten[i].ausgeben(out);
    }
    out << endl;

    Note durchschnitt = berechneDurchschnitt();
    out << "Durchschnitt : ";
    durchschnitt.ausgeben(out);
}
```

Das gibt schon eine recht ansprechende Ausgabe. Wir können die Methode ausgeben der Liste von einem main aus aufrufen und dieser einfach cout übergeben. Im Beispiel erzeuge ich aber auch eine Datei! Da die Klasse ofstream sehr ähnlich ist wie das cout – Objekt können wir der Methode asugeben einfach auch das file – Objekt mitgeben. Hier das Beispiel :

```
#include "Notenliste.h"

#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    Notenliste liste("C++");

    for(long i = 0; i < 10; ++i)
    {
        double testNote = 4.0 + (double)i / 10.0;
        Note note(testNote);
        liste.hinzufuegen(note);
    }

    liste.ausgeben(cout);

    // Datei erzeugen
    ofstream out("Noten.txt");

    liste.ausgeben(out);

    return 0;
}
```

Diese Datei lässt sich sogar in Excel importieren. So können wir überprüfen, ob unsere Durchschnittsberechnung stimmt.

Tipps zum Runden

Das Runden scheint nicht so einfach zu sein. Ich kann im Moment nicht einmal sagen, ob die C++ Library solche Funktionen zur Verfügung stellt. Sie sind aber nicht so schwer selber zu schreiben!

Beispiel Zehntel

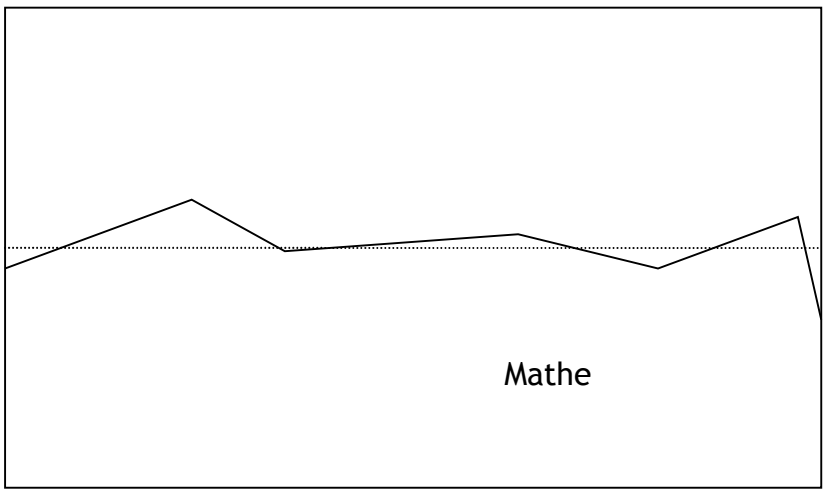
Rechne den Notenwert mal 10. Weise diesen Wert einer Ganzzahl zu, dadurch verschwinden die Nachkommastellen. Nun teile diese Zahl wieder durch 10, aber achte darauf, dass es eine genaue Rechnung gibt (mit doubles). Um bei den Rundungsgrenzen auch richtig zu liegen müssen wir zur Note soviel addieren, dass die Note, die gerade noch aufgerundet werden kann (hier x.x5) den nächsten zehntel erreicht. Bei viertel Noten oder halben Noten ist dieser Korrekturwert anders!

```
double note = 5.25;
double ergebnis = 0.0;
int temp = (int)(10.0 * (note+0.05));
ergebnis = (double)temp / 10.0;
```

Die anderen Rundungswerte sollten auf ähnliche Art zu berechnen sein!

Grafische Ausgabe

Die neueste Version der Zeichenflaeche bietet also die Möglichkeit Linien auszugeben. Ich stelle mir etwa folgende Grafik vor :



Eine Möglichkeit eine solche Grafik zu erzeugen ist es, der Notenliste eine Zeichenflaeche als Datenelement hinzuzufügen. Danach gilt es die Zeichenflaeche vernünftig aufzuteilen, das heisst die X-Achse und die Y-Achse optimal auszunutzen, janach Anzahl Noten die sich in der Notenliste befinden.
Hole Dir die neueste Version von TsuZeichnen.exe vom Internet und die passenden Klassen : ZeichenFlaeche, Kreis, Linie, Text.