Hochschule Darmstadt
Prof. Dr. Benjamin Meyer

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

16. April 2021

Programmieren, Algorithmen und Datenstrukturen II, SoSe 2021
Aufgabenblatt 1

1 Einleitung

Willkommen bei Ihrem ersten Praktikum zu Programmieren, Algorithmen und Datenstrukturen II. Neben den Inhalten der ersten Vorlesungseinheit (Streams) soll dieses Aufgabenblatt vor allem der Wiederholung der in PAD1 erlernten Konzepte dienen.
Falls aus PAD1 nicht schon vorhanden, benötigen Sie für das Praktikum eine lauffähige, korrekt eingerichtete *integrierte Entwicklungsumgebung*. Es gibt viele kostenfreie gute Alternativen, für diesen Kurs empfehlen wir jedoch die IDE Qt, bei Problemen werfen Sie bitte einen Blick auf das hochgeladene PDF *Installationsanleitung_Qt.pdf*.
Wichtig ist zudem die Installation eines Compilers. Installieren Sie daher zusammen mit Qt die Compilersuite MinWG. Wie in dem pdf gezeigt, können Sie in dem Installationsschritt *Komponenten auswählen* von Qt unter *Qt 5.15.1* den Unterpunkt *MinWG 8.1.0* anwählen (das sind die derzeit aktuellen Versionen, kann natürlich variieren). Wählen Sie 32 oder 64 Bit entsprechend Ihrer Rechner-Architektur. MinWG lässt sich jedoch auch separat installieren.
Sollte die Qt-Installation einfach nicht gelingen, lohnt es sich vielleicht, einen Blick auf andere IDEs wie CLion von JetBrains oder Visual Studio von Microsoft zu werfen (vor allem für Mac-User empfehlenswert, hier scheint die Installation von Qt nicht ganz trivial).

1.1 Video Streaming Datenbank

In dieser Aufgabe soll ein Streaming-Dienst für Filme ins Leben gerufen werden. Hierfür wird eine Datenstruktur zur Speicherung und Verwaltung der einzelnen, zur Verfügung stehenden Titel benötigt. Sie müssen daher zwei Klassen erstellen.
Eine Klasse `Movie` zur Repräsentation eines einzelnen Videos mit folgenden Attributen und Methoden (Sie dürfen gerne sinnvoll ergänzen):

- Attribute
  - Titel
  - Länge
  - Liste von Bewertungen (Skala 1-5 Punkte)
  - Durchschnittliche Bewertung
  - Genre
- Methoden
  - Konstruktor
  - Ausgabe (aller wichtigen Informationen)
  - Hinzufügen einer Bewertung (Skala 1-5 Punkte)
  - Video abspielen (geben Sie hierfür einfach einen String auf der Konsole aus)

Eine Klasse `Database`, die die Videos verwaltet

- Attribute
  - Vektor der Videos
- Methoden
  - Ausgabe aller Videos
  - Hinzufügen eines Videos
  - Entfernen eines Videos (anhand der Position in der Liste)
  - Sortieren der Liste nach absteigendem Bewertungsgrad (Sie können hierfür eine Standardbibliothek nutzen, oder einen bekannten Sortier-Algorithmus selber implementieren)
  - Ausgabe der durchschnittlichen und gesamten Abspieldauer aller Videos
  - Abspielen eines Videos (anhand der Position in der Liste)

Da uns eine einzelne Datenbank zur Verwaltung der Videos reichen soll, ist es nicht nötig, ein Objekt der Klasse anzulegen. Deklarieren Sie daher einen einzigen, privaten Konstruktor, um Instanziierung der Klasse zu verhindern, und gestalten Sie die geforderten Methoden sowie den Video-Vector statisch. Testen Sie Ihre Methoden in der `main()`, um die korrekte Funktionalität sicherzustellen.

1.2 Einlesen einer Datenbank

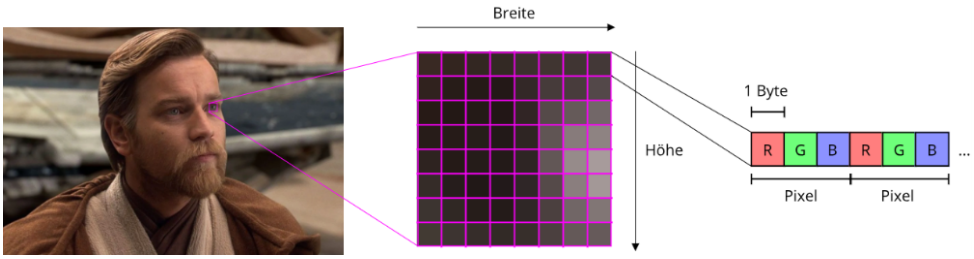
Damit bei Programmstart nicht alle Titel neu eingegeben werden müssen, soll in diesem Schritt eine statische Methode `init()` geschrieben werden, die eine bestehende Datenbank (in Form der im Moodle zu findenden Datei `database.txt`) einliest und in die entsprechende Dateistruktur des Streaming-Dienstes integriert. Nutzen Sie die Referenzseite für C++ (`https://www.cplusplus.com/reference/`), um mit den Möglichkeiten des `ifstream` und des `ofstream` vertraut zu werden.

1.3 Video-Bewertung und Schreiben der Datenbank

Abschließend soll eine Bewertung der Videos durch alle Nutzer simuliert werden. Die statische Methode `simulateReviews(int numberOfReviews)` soll insgesamt `numberOfReviews`-mal einem zufälligen Video eine ebenso zufällig generierte Bewertung hinzufügen. Sortieren Sie nun die Videos nach der neuen Durchschnittsbewertung und speichern Sie die Videos danach in dieser Reihenfolge in einer neuen txt-Datei ab.

1.4 Bonusaufgabe: Bilder lesen, schreiben und bearbeiten

Keine Filmesammlung wäre komplett ohne Filmposter. In dieser Bonusaufgabe sollen Sie ein Stück Code schreiben, welches Bilder von Filmpostern einlesen, bearbeiten und schreiben kann. Zunächst stellt sich hierfür die Frage: Was sind eigentlich Bilder in der Informatik? Für unsere Zwecke ist ein Bild nichts anderes als ein zweidimensionales Array von Einträgen, die wir 'Pixel' nennen. Ein Bild mit den Ausmaßen 'Breite' * 'Höhe' besteht dabei aus 'Höhe' Zeilen mit jeweils 'Breite' Pixeln. Ein Pixel in einem typischen Farbbild besteht dabei aus 3 separaten Werten die den Rot-, Grün- und Blauanteil definieren. Diese Anteile werden üblicherweise als Zahlenwerte im Bereich [0;255] angegeben, wobei der Wert 0 für keine Intensität ('schwarz') und der Wert 255 für volle Intensität steht. Das folgende Bild veranschaulicht diese Zusammenhänge:



Alle modernen Bildformate, wie z.B. JPEG, PNG oder TIFF, sind in ihrer Struktur relativ komplex, daher nutzen wir ein einfaches Bildformat namens PPM . Eine PPM-Datei besteht aus zwei Teilen: Einem kleinen Header, welcher Informationen über das Bild enthält, sowie einen Block mit den Pixel-Daten:

```
1  P6
2  800 1200
3  255
   □□$%) !%&&(**,□□!<<>]]]88.....
```

Header (Text)

Pixel (Binär)

Der Header besteht aus drei Text-Zeilen. Die erste Zeile enthält eine Konstante ('P6') welche das Dateiformat identifiziert. In Zeile 2 stehen die Breite und Höhe des Bildes in Pixeln. Zeile 3 enthält den Maximalwert der Intensität für alle Pixel. Der Header endet mit einem einzelnen Leerzeichen. Die Pixel-Daten liegen in Binärform vor, Zeile auf Zeile beginnend an der oberen linken Ecke des Bildes. Dabei besteht jeder Pixel aus 3 Werten vom Typ unsigned char für die Rot-, Grün- und Blaukomponenten, wie im ersten Bild dargestellt.

Nun zur eigentlichen Aufgabe:

Teil 1

Schreiben Sie ein C++-Programm welches die dem Aufgabenblatt beiliegende Bilddatei `cover1.ppm` einliest und in eine andere Datei wieder herausschreibt. Lesen Sie dazu zunächst den Header der PPM-Datei ein und extrahieren Sie die Information über Breite und Höhe der Datei. Geben Sie diese auf der Konsole aus. Lesen Sie dann alle Pixel in einen vector ein. Anschließend schreiben Sie den Header gefolgt von den Pixeln in eine neue Datei. Vergleichen Sie mit einem Bildbearbeitungsprogramm (z.B. GIMP), dass die neu geschriebene Datei korrekt aussieht.

Tipps:

a) Der Header liegt im Textformat vor, kann also mit Hilfe des >> Operators ausgelesen werden und mit Hilfe des << Operators in die neue Datei geschrieben werden.

b) Die gleiche Vorgehensweise ist für die Pixel nicht möglich. Verwenden sie hierfür die im Vorlesungsskript vorgestellten Funktionen zum Lesen von Binärdaten

c) WICHTIG: Der Header wird mit einem einzelnen Leerzeichen abgeschlossen. Nutzen sie die im Vorlesungsskript vorgestellte Funktion zum Ignorieren einzelner Zeichen in einem Stream

d) Die einzelnen Pixel-Werte liegen zwar als `unsigned char` Werte vor, für den nächsten Aufgabenteil bietet es sich aber an, eine eigene Klasse anzulegen, welche die Rot-, Grün- und Blauwerte für einen Pixel kapselt

Teil 2

Wenn Sie die Pixel-Daten in einen `vector` eingelesen haben können Sie diese bearbeiten. Eine einfache Operation ist das Umwandeln eines Farbbildes in ein Graustufenbild. Ein Graustufenbild zeichnet sich dadurch aus, dass die Rot-, Grün- und Blauwerte eines Pixels identisch sind. Zur Berechnung des Grauwerts eines Pixels können Sie folgende Formel verwenden:

$$Grauwert = 0.289 * rot + 0.587 * grn + 0.114 * blau \tag{1}$$

Wenden Sie diese Formel für jeden Pixel im Bild an und setzen Sie anschließend den Rot-, Grün- und Blauwert des Pixels auf den errechneten Grauwert. Schreiben Sie anschließend das Graustufenbild in eine Datei und überprüfen Sie es mit einem Bildbearbeitungsprogramm.