UNIVERSITÄT BAYREUTH

Lehrstuhl für Angewandte Informatik III
(Robotik und Eingebettete Systeme)
Prof. Dr. Dominik Henrich

Bachelorpraktikum AI3

Aufgabe 1: Einlesen einer Textdatei

Gegeben ist eine Textdatei mit biologischen Daten verschiedener Kreaturtypen. Jeder Eintrag umfasst eine Zeile. Die einzelnen Daten (Felder) eines Kreaturtyps sind durch einen Komma getrennt. Vor dem ersten Feld und nach dem letzten Feld steht kein Komma. Es kann leere Felder geben (erkennbar durch zwei aufeinanderfolgende Komma oder einen Komma am Anfang bzw. Ende der Zeile).

Jede Zeile enthält die folgenden Felder in der angegebenen Reihenfolge:

```
Kreaturname,Stärke,Geschwindigkeit,Lebensdauer,Eigenschaften,Bild
```

Die Felder genügen den folgenden Beschränkungen:

- `Kreaturname` darf nur normale Buchstaben sowie Leerzeichen enthalten und muss angegeben werden.
- `Stärke`, `Geschwindigkeit`, und `Lebensdauer` müssen positive Ganzzahlen sein.
- `Eigenschaften` beinhaltet beliebig viele, leerzeichen-getrennter Wörter, aus den Buchstaben a-zA-Z, Unterstrich, und Zahlen. Die Reihenfolge der Eigenschaften ist beliebig.
- `Bild` muss ein relativer Dateipfad nach POSIX-Standard sein.
- Für weitere Beispiele von fehlerhaften oder zulässigen Kreaturtypen siehe die beiden Eingabedateien CreatureTable.txt und CreatureTable_mitFehlern.txt, die im eLearning verfügbar sind.

Schreiben Sie ein Programm zum Einlesen einer Datei im gegebenen Format. Sind Fehler in einer Zeile, soll diese Zeile ignoriert werden, eine entsprechende Meldung über den Fehlerort und -typ auf der Konsole ausgegeben werden und die Datei weiter eingelesen werden bis zu ihrem Ende. Treten Fehler beim Öffnen der Datei auf, so soll auch dies gemeldet werden.

Den Dateinamen zum Einlesen erhält Ihr Programm als ersten und einzigen Parameter auf der Kommandozeile in der main-Methode (siehe argc, argv).

Geben Sie nach dem Einlesen statistische Daten an, zumindest die Zahl der insgesamt eingelesenen Zeilen und die Zahl der korrekten und fehlerhaften Zeilen.

Ebenso sollen die Daten nach dem Einlesen in einer geeigneten Datenstruktur im Hauptspeicher für die Weiterverarbeitung abgelegt werden. Diese Datenstruktur sollte insbesondere ihre Elemente (also einzelne Kreaturtypen) selbständig verwalten, und automatisch löschen.

Der Code ist zweiteilig zu dokumentieren: Eine Header-Dokumentation für Klassen und Funktionseinstiegspunkte, sowie eine Source-Dokumentation für Fallstricke in

der Implementierung. Triviale Dokumentation ist nicht erwünscht, die Dokumentation sollte auf Codeblöcken statt auf Codezeilen gründen.

Nach Abschluss der Programmierarbeiten ist der Code aufzuräumen. Insbesondere ist eine konsistente Konvention für Variablen-, Funktions- und Klassennamen, für Einrückung, Klammernsetzung, sowie Whitespace unerlässlich. Unlesbarer oder unübersichtlicher Code wird nicht akzeptiert.

Hinweise:

- Bereits für diese Aufgabe ein strukturiertes Projekt aufzusetzen ist nicht nötig, spart später aber viel Arbeit.
- Verwenden Sie zum Umgang mit Daten und Dateien nach Möglichkeit die Klassen und Methoden der Standardbibliothek (vector, list, fstream, getline, string, stringstream).
- Achten Sie beim Anlegen Ihrer Klassen und Datenstrukturen auf eine saubere Architektur. Insbesondere soll das Hinzufügen neuer Spalten bzw. das Ändern von Spaltenbeschränkungen ohne viel Aufwand realisierbar sein.
- Achten Sie auf korrektes C++. Dazu gehören korrekter Umgang mit Speicher, Ausnahmen, und Stackvariablen. Speicherlecks sind zu vermeiden.
- Achten Sie auf sinnvolles Klassendesign: Member-Methoden, private/public-Spezifikationen, statische Lademethoden, Ausnahmen für Fehler.
- Sinnvolle Google-Stichwörter: STL, vector, list, destructor, constructor, const reference, pass-by-value, pass-by-reference, exception safety