Abteilung für
Angewandte Mathematik

SS 2024
Ludwig Striet

Einführung in die Programmierung für Studierende der Naturwissenschaften

Blatt 6 – 03.06.2024

Abgabe: *Bis 09.06.2024, 23:59 Uhr per E-Mail an Ihre/n Tutor/in*

Aufgabe 1 (5 Punkte)**.** In dieser Aufgabe geht es darum, einen (sehr einfachen) Vokabeltrainer zu implementieren. Schreiben Sie ein Programm, das das folgende tut:

- Das Programm soll die Datei `vokabeln.txt` einlesen und in einem `std::vector<std::string> vokabeln` speichern. In der Datei steht pro Zeile ein Wordpaar in der Form `<englisch>\t<deutsch>`. Das englische und deutsche Wort sind dabei durch ein Tabulator-Symbol `'\t'` getrennt, das im Texteditor möglicherweise als Leerraum angezeigt wird.
- Danach soll das Programm zufällige Einträge aus dem `std::vector` auswählen, das Englische Wort ausgeben und nach dem passenden Deutschen Wort fragen. Das Programm soll ausgeben, ob die Übersetzung korrekt war oder nicht. Die korrekte Übersetzung soll nicht ausgegeben werden. Es sollen solange Vokabeln abgefragt werden, bis zehn mal hintereinander kein Fehler gemacht wurde.

Aufgabe 2 (5 Punkte)**.** In dieser Aufgabe geht es darum, eine Liste mit Vokabeln zu erstellen, die dann von dem Programm aus Aufgabe 1 gelesen werden können. Schreiben Sie dazu ein Programm, das das folgende tut

- Es soll als erstes ein Dateiname `std::string dateiname` abefragt werden. Falls die Datei schon existiert soll sie gelesen und die Einträge in einem `std::vector<std::string> vokabeln` gespeichert werden, sonst soll der `vokabeln` zu Beginn leer sein.
- Danach sollen nacheinander englische Wörter sowie die Deutsche Übersetzung eingegeben werden. Wenn der Eintrag `<englisch>\t<deutsch>` noch nicht in `vokabeln` vorhanden ist, soll er am Ende eingefügt werden. Das Programm soll nach jedem eingegebenen Wortpaar abfragen, ob ein weiteres eingegeben werden soll.
- Wenn kein weiteres Wortpaar eingegeben werden soll, sollen die Vobabeln in der Datei `dateiname` gespeichert werden. Falls die Datei schon existiert soll der bisherige Inhalt überschrieben werden.

Hinweise (Aufgabe 1 und 2), bitte auch Rückseite beachten: (i) ein `std::string`, der ein Leerzeichen enthalten kann, kann vom Terminal so eingelesen werden:

```
std::string s;
std::getline(std::cin, s);
```

(ii) es gibt natürlich viele Sonderfälle, wie beispielsweise, dass ein Wort mehrere gültige Übersetzungen hat etc. Diese müssen Sie nicht berücksichtigen.

(iii) Sie können Ihre Implementierungen der `read_list(...)`, `write_list(...)` und `split_at_tab(...)` Funktionen vom letzten Blatt verwenden. Alternativ finden Sie Implementierungen auf der Vorlesungswebseite.

Aufgabe 3 (10 Punkte)**.** In dieser Funktion soll eine einfache Klasse für ein Frachtschiff implementiert werden. Ein Frachtschiff hat folgende Eigenschaften:

- Einen Namen, der sich nie ändert
- Eine maximale Ladungsmenge, die sich nie ändert
- Eine aktuelle Ladung, die zu Anfang gleich 0 ist und immer größer als 0 und kleiner oder gleich der maximalen Ladung ist.

Außerdem hat ein Frachtschiff Methoden um

- Seinen Namen abzurufen
- Die aktuelle Ladungsmenge und die maximale Ladungsmenge abzurufen
- Eine bestimmte Menge zu beladen. Dabei wird überprüft, dass die maximale Ladung nicht überschritten wird. Andernfalls belädt das Schiff nichts und gibt eine Fehlermeldung aus.
- Eine bestimmte Menge zu entladen. Dabei wird überprüft, dass nicht mehr entladen wird als das Schiff momentan geladen hat. Andernfalls wird nichts entladen und eine Fehlermeldung ausgegeben.

Implementieren Sie eine Klasse `Frachtschiff`, die die geforderte Funktionalität erfüllt. Speichern Sie Ihre Implementierung zusammen mit einer `main`-Methode, in der Sie die Funktionalität demonstrieren, in einer Datei `frachtschiff.cc`.