IMS
Institut für Mikroelektronische Systeme
Leibniz Universität Hannover

Programmierpraktikum Technische Informatik (C++)
Aufgabe 02

Hinweise
Abgabe: Stand des Git-Repositories am 5.5.2020 um 9 Uhr.
Die Dateien zur Bearbeitung dieser Aufgabe erhalten Sie, indem Sie die neue Aufgabe aus dem Aufgabenrepository in Ihr lokales mergen. Dies geschieht mit `git pull cpp2020 master` innerhalb Ihres Repositories. Die Lösungen committen Sie bitte in Ihr lokales Repository (`git commit -a` oder `git add` gefolgt von `git commit`) und pushen sie in Ihr Repository auf dem git-Server des Instituts (`git push`).

Teilaufgabe 1 (2 Punkte)

In der Datei `nutriScore/src/nutri.cpp` sollen Funktionen implementiert werden, die Lebensmittel anhand eines Nutri-Scores und der verzehrten Menge analysieren. Es gilt, die Anzahl der gesunden Lebensmittel zu bestimmen, welche hier anhand ihrer Nährwerte eingeordnet werden, sowie die ungesündeste Essgewohnheit zu finden. Die Lebensmittel werden in Form des Tupels `std::tuple<std::string, char, size_t>` verwaltet. An erster Stelle im Tupel steht immer der Name des Lebensmittels, an der zweiten Stelle der Nutri-Score und an der letzten die Anzahl, wie häufig das jeweillige Lebensmittel verzehrt wurde. Der Nutri-Score ist ein System zur Nährwertkennzeichnung, welches die Buchstaben A bis E verwendet, wobei A für viele gute und E für viele schlechte Inhaltsstoffe steht.

a) Implementieren Sie die Funktion `getUnhealthy`!
Diese Funktion soll entscheiden, welche der beiden ungesunden Lebensmittel `mealA` bzw. `mealB` das ungesündere für den Verbraucher war und dieses dann zurückgeben. Die Funktion muss nur Lebensmittel mit dem Score D und E vergleichen können. Das Lebensmittel, welches häufiger gegessen wurde zählt als ungesünder, wobei Gerichte mit dem Score E doppelt zählen. Somit ist ein Lebensmittel mit dem Score E, welches 3 mal gegessen wurde, ungesünder als ein Lebensmittel mit dem Score D, welches 5 mal verzehrt wurde.

Hinweise:

- Sie können `std::get<n>` verwenden, um auf die einzelnen Elemente eines Tupels zuzugreifen. Die Dokumentation dieser Funktion können Sie unter `http://en.cppreference.com/w/cpp/utility/tuple/get` nachlesen.

Appelstraße 4 | 30167 Hannover
fon 0511 762-19691 | fax 0511 762-19694
www.ims.uni-hannover.de | info@ims.uni-hannover.de

- Es kann davon ausgangen werden, dass der Nutri-Score immer in Großbuchstaben angegeben wird

b) Implementieren die Funktion `analyzeMeals`!
Diese erhält als Parameter `meals` einen Vector aller verspeisten Lebensmittel, die analysiert werden sollen. Die Ergebnisse sollen in den übergebenen `std::ostream&` geschrieben werden. Die Methode soll angeben, wie viele verschiedene Lebensmittel insgesamt verzehrt wurden und wie viele davon als gesund gelten. Zudem soll die schlechteste Essgewohnheit in Form des ungesündesten Lebensmittels ermittelt und ausgegeben werden.

Hinweise:

- Lebensmittel mit dem Score A oder B gelten als gesund, E und D gelten als ungesund
- Verwenden sie zum Vergleichen der ungesunden Lebensmittel die Funktion `getUnhealthy`
- Jedes Lebensmittel befindet sich nur jeweils einmal in dem Vector
- Falls kein ungesundes Lebensmittel in dem Vector vorhanden ist, soll dies anstelle des ungesündesten Lebensmittels ausgegeben werden

Teilaufgabe 2 (1 Punkt)

Implementieren Sie in der Datei `csvParser/src/parser.cpp` folgende Funktionen, um Eingabedaten anhand von Trennzeichen aufzusplitten!

a) Implementieren Sie die Funktion `split(std::istream& is, char delim)`!
Diese Funktion soll aus dem übergebenen `std::istream` durch das übergebene Trennzeichen `delim` getrennte Strings auslesen und die eingelesenen Strings in einem `std::vector` zurückgeben. Leere Strings, die beispielsweise entstehen, wenn mehrere Trennzeichen aufeinander folgen, sollen im Ergebnis-Vektor nicht vorkommen.

Hinweise:

- Sie können `std::getline` verwenden, um einen String bis zum angegebenen Trennzeichen einzulesen. Die Dokumentation dieser Funktion können Sie unter `http://en.cppreference.com/w/cpp/string/basic_string/getline` nachlesen.
- Lesen Sie in einer Schleife solange Strings ein, wie `is` noch Zeichen enthält.
- Hier kann der Rückgabewert von `getline` verwendet werden.

Teilaufgabe 3 (1 Punkt)

Implementieren Sie die Funktion `parse(std::istream& is)`! Die Funktion soll eine CSV-Datei einlesen und in einen Vector von IndexedStrings umwandeln. Die Datenstruktur IndexedString ist in `parser.h` vorgegeben. Es handelt sich um einen Tuple eines Integers und eines Strings. Der Integer-Wert steht dabei für die Position des Strings in einem Text.

Hinweise:

- Beachten Sie, dass Einträge einer CSV-Datei durch ein bestimmtes Zeichen voneinander getrennt sind. In diesem Fall ist es das Zeichen `';'`. Die Funktion `split` aus Teilaufgabe 1 könnte sich zum Trennen nach diesem Zeichen als nützlich erweisen.
- Eine gültige CSV-Datei, die von dem Parser eingelesen werden können soll, hat folgende Merkmale:
  - Die Datei hat eine Menge von Einträgen, die durch ein `';'` getrennt sind.
  - In abwechselnder Folge sind die Einträge Nummern und Strings.
  - Der erste Eintrag ist eine Nummer (der Index).
  - Ein IndexedString setzt sich aus einer Nummer und dem darauf folgenden String zusammen.
- Sie können davon ausgehen, dass nur gültige CSV-Dateien eingelesen werden. Fehlerfälle müssen nicht berücksichtigt werden.
- Die Funktion `split` gibt eine Zahl aus der Datei als String zurück. Um diesen in einen Integer zu wandeln, können Sie die Funktion `std::stoi` verwenden. Die Dokumentation dieser Funktion können Sie unter `http://en.cppreference.com/w/cpp/string/basic_string/stol` nachlesen.

Teilaufgabe 4 (1 Punkt)

Schreiben Sie die Funktion `writeSentence`! Diese soll den übergebenen Vektor von IndexedStrings aus Teilaufgabe 2 in sortierter Reihenfolge in den übergebenen `std::ostream` schreiben. Die Reihenfolge ist dabei mit aufsteigendem Index bei 0 beginnend. Sie können davon ausgehen, dass die Index-Nummern lückenlos sind und keine Nummer doppelt vorkommt. Beim Schreiben in den `std::ostream` soll nur der String ausgegeben werden, nicht der Index.

Hinweis: Auf einzelne Bestandteile eines IndexedString kann mit der Funktion `std::get<n>` zugegriffen werden. Die Dokumentation dieser Funktion können Sie unter `http://en.cppreference.com/w/cpp/utility/tuple/get` nachlesen.