Programmiertechnik II

Sommersemester 2015 – Aufgabenblatt 3

Hasso-Plattner-Institut
Fachgebiet Computergrafische Systeme

Anmerkungen zu Aufgabenblatt 3

Der zu verwendende Programmrahmen steht als Zip-Archiv `uebung3.zip` im Moodle unter der Adresse `https://moodle.hpi3d.de/course/view.php?id=75` zum Download zur Verfügung.

Die Aufgaben sollen in den benannten und im Programmrahmen zumeist gekennzeichneten Bereichen implementiert werden. Bevor Sie mit der Lösung beginnen, lesen Sie das gesamte Aufgabenblatt und überprüfen Sie, ob der Programmrahmen auf Ihrem System fehlerfrei kompiliert, die Datensätze enthalten sind und sich alle Programme ausführen lassen. Wir wünschen viel Spaß und Erfolg bei der Bearbeitung der Aufgaben!

Aufgabe 3.1: Manipulation von Zeichenketten (8 Punkte $^{2+2+4}$)

In dieser Aufgabe sollen E-Mail-Adressen auf ihre Plausibilität überprüft werden. Die E-Mail-Adressen und die zugehörigen Informationen (Name, Vorname, Firma) sind in einem Array definiert. Implementieren Sie im Programmrahmen die Funktionen `caseConvert()`, `replace()` und `eMailCheck()` in der Datei `emailcheck.cpp` wie folgt:

- Die Funktion `caseConvert()` soll in einer Zeichenkette eventuell vorhandene Großbuchstaben durch kleine Buchstaben ersetzen.
- Die Funktion `replace()` soll in einer Zeichenkette die Umlaute ä, ö, ü sowie ß durch ae, oe, ue sowie ss ersetzen.
- Die Funktion `emailCheck()` soll eine als Zeichenkette gegebene E-Mail Adresse (vorname.nachname@firma.de) auf Plausibilität in Bezug auf die Tokens der Strings prüfen.
  Beispiele:
  ```
  emailCheck("juergen.doellner@hpi.de", "Jürgen", "Döllner", "HPI") => true
  emailCheck("sebastian.deissler@bmw.de", "Paul", "Deißler", "BMW") => false
  emailCheck("mueller.marga@sap.com", "Marga", "Müller","SAP") => true
  emailCheck("h.boss@service.bayer.com", "Hugo", "Boss", "Bayer") => true
  ```

Aufgabe 3.2: Exception-Handling (10 Punkte $^{2.5+4+1.5+2}$)

Um die Stabilität eines Programms gegenüber ungültigen Eingaben oder Fehlern während der Programmausführung zu erhöhen, können an kritischen Code-Stellen Exceptions geworfen und in sinnvoller Weise behandelt werden. In dieser Aufgabe soll ein gegebenes Programm zur Verifikation von Wetterdaten um entsprechendes Exception-Handling erweitert werden. Das Modul `exceptions.cpp` liest einen gegebenen Datensatz (`weather_babelsberg.csv`) ein, der tagesgenaue Temperatur und Niederschlagswerte enthält. Der Dateipfad und -name wird als Kommandozeilenargument übergeben. Vergegenwärtigen Sie sich zunächst den Datensatz in einem Texteditor. Machen Sie sich zudem mit der Struktur `FormatException` im Programmrahmen vertraut.

a) Implementieren Sie die Funktion `checkData()`, die eine CSV-Datei öffnet, zeilenweise einliest und pro Zeile die Funktion `parseLine()` aufruft. Fangen Sie Exceptions, die beim Öffnen, Lesen oder Schließen von Dateien üblicherweise auftreten, ab. Machen Sie sich hierzu mit der Methode `std::ios::exceptions` vertraut.

b) Implementieren Sie die Funktion `parseLine()`, die eine gegebene Zeile elementweise (getrennt durch einen Delimiter) zerlegt. Nutzen Sie die Funktionen `stringToTime()` und `std::stof()` in Verbindung mit Exception-Handling um die Validität der verschiedenen Daten-Felder zu überprüfen. Bei fehlerhaften Zeilen, soll eine entsprechend initialisierte `FormatException` geworfen werden.

c) Erweitern Sie die Funktion `checkData()` indem Sie `FormatException`s abfangen. Rufen Sie im entsprechenden catch-Block die Methode `writeOutFormatException` auf. Ermitteln Sie zudem die Anzahl an validen und invaliden Einträgen in der CSV-Datei und geben Sie diese Werte auf der Konsole aus.

d) Implementieren Sie die Funktion `writeOutFormatException()`, die eine `FormatException` in einen Logfile exportieren soll. Pro Exception sollen dabei die Nummer der betroffenen Zeile der CSV-Datei und die invaliden Daten-Felder (Date, Temperature oder Rainfall) ausgegeben werden. Fangen Sie Exceptions, die beim Öffnen, Schreiben oder Schließen von Dateien auftreten können ab.

Aufgabe 3.3: Spürhunde (3 Punkte)

Für das Training von Spürhunden befüllt ein Trainer n Boxen mit unterschiedlichen Gegenständen und stellt sie in einer Reihe auf. Die Art eines Gegenstandes ist durch eine positive Zahl gekennzeichnet. Zum Trainieren eines Hundes lässt der Trainer diesen an Gegenstand x schnuppern und schickt ihn von einem der beiden Enden der Reihe los. Der Hund überprüft nun der Reihe nach alle Boxen auf der Suche nach einer Box mit einem x-Gegenstand. Hat der Hund eine solche Box gefunden, bringt er sie dem Trainer unverzüglich. Findet der Hund den Gegenstand nicht, kehrt er ohne Box zurück.

Nun möchte der Trainer zwei Hunde gleichzeitig trainieren indem er beide gleichzeitig losschickt. Damit die Hunde sich nicht um eine Box streiten, müssen beide nach unterschiedlichen Gegenständen suchen. Dazu kann der Trainer die Hunde entweder von beiden Seiten der Reihe (links und rechts) oder beide Hunde von einer Seite (links oder rechts) gleichzeitig starten lassen. Die Hunde benötigen genau eine Sekunde um an einer Box zu schnüffeln, jedoch laufen sie so schnell, dass ihre Laufzeit vernachlässigt werden kann. Der Trainer will nun wissen, wie viele Sekunden er mindestens als Suchzeit einplanen muss.

Implementieren Sie die Methode `minTimeDogTraining()` im Modul `sniffer_dog.cpp`. Die Funktion soll für eine gegebene Suchkombination die minimale Anzahl von Sekunden zurückgeben, die der Trainer warten muss, bzw. -1 wenn es keine solche Kombination gibt.

Beispiel:
Der Aufruf `minTimeDogTraining(4, 5, std::vector<int>{1,4,3,2,5})` soll das Ergebnis 2 liefern.

Allgemeine Hinweise zur Bearbeitung und Abgabe

- Die Aufgaben sollen maximal zu zweit bearbeitet werden; Ausnahmen sind nicht vorgesehen. Je Gruppe ist nur eine Abgabe notwendig.
- Bitte reichen Sie Ihre Lösungen bis **Montag, den 18. Mai um 09:00 Uhr** ein.
- Zur Prüfungszulassung muss ein Aufgabenblatt zumindest bearbeitet werden und alle weiteren Aufgabenblätter mit mindestens 50% der Punkte bewertet werden. Aufgaben mit Klausurpunkten werden 1:1 berücksichtigt.
- Die Aufgaben können auf allen wesentlichen Plattformen (Windows, Linux, OS X) bearbeitet werden. Lesen Sie bei Fragen zum Kompilieren und Ausführen bitte genau die den Archiven jeweils beiligende README.TXT. Bestehen weitere Fragen und Probleme, kontaktieren Sie einen der Betreuer direkt oder nutzen Sie das Forum im Moodle.
- Die Durchsicht zur Bewertung kann sowohl auf Windows, Linux oder OS X erfolgen. Es wird folglich eine **plattformunabhängige Lösung** der Aufgaben erwartet.
- Archivieren (Zip) Sie zur Abgabe Ihren bearbeiteten Programmrahmen und ergänzen Sie Ihre Matrikelnummer(n) im Bezeichner des Zip-Archivs entsprechend im folgenden Format: **`uebung3_matrikNr1_matrikNr2.zip`**. Geben Sie die schriftlich bearbeitete Aufgabe 2.1 als Textdokument (.txt) oder PDF im Archiv hinzugefügt ab. Beachten Sie, dass dabei nur die vollständigen Quelltexte, die CMake-Konfiguration sowie eventuelle Zusatzdaten gepackt werden (alles was im gegebenen Programmrahmen entsprechend vorhanden ist). Senden Sie uns keinesfalls Kompilate und temporäre Dateien (*.obj, *.pdb, *.ilk, *.ncb etc.). Testen Sie vor dem Verschicken, ob die Projekte aus den Zipdateien fehlerfrei kompiliert und ausgeführt werden können.
- Reichen Sie Ihr Zip-Archiv im Moodle ein:
`https://moodle.hpi3d.de/course/view.php?id=75`.