Programmiertechnik II

Sommersemester 2017 – 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=96` zum Download zur Verfügung.

Schriftlich zu bearbeitende Aufgaben reichen Sie bitte als PDF-Datei ein, die Sie dem Zip-Archiv der Abgabe hinzufügen.

Aufgabe 3.1: Exception-Handling (10 Punkte $^{2+4+2+2}$)

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 Datenfelder 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 ein 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 ab, die beim Öffnen, Schreiben oder Schließen von Dateien auftreten können.

Aufgabe 3.2: Algorithmen und Funktionsobjekte in C++ (6 Punkte $^{2+2+2}$)

Die Programmierung von Algorithmen wird in C++ durch Lambda-Ausdrücke unterstützt. In dieser Aufgabe sollen die zugehörigen Programmiersprachmittel eingesetzt werden. Zwei Datensätze (`airports.dat` und `routes.dat`), die Informationen über Flughäfen bzw. Informationen über regelmäßige Verbindungen zwischen Flughäfen enthalten, sollen durch entsprechende Algorithmen ausgewertet werden. Das Modul `airports.cpp` liest beide Datensätze ein und speichert alle für die Aufgabe relevanten Informationen über einen Flughafen in einer `AirportInfo`-Struktur. Die `AirportInfo`-Strukturen werden wiederum der entsprechenden Flughafen-ID zugeordnet und in einer `std::map` gespeichert.

a) Implementieren Sie die Funktion `removeNonDirectFlights`, die aus dem `std::vector AirportInfo::m_routes` alle Routen entfernt, die mindestens einen Zwischenstopp beinhalten. Nutzen Sie dafür unter anderem die Funktion `std::remove_if`.

b) Implementieren Sie die Funktion `calculateDistancePerRoute`, die für jede Route aus `AirportInfo::m_routes` die Distanz zwischen Start- und Zielflughafen in Kilometern berechnet. Nutzen Sie unter anderem die Funktion `std::transform`. Machen Sie sich mit der im Programmrahmen gegebenen Funktion `calculateDistanceBetween` vertraut. Speichern Sie das Ergebnis in `AirportInfo::m_routeLengths`.

c) Implementieren Sie die Funktion `calculateAverageRouteDistances`, die pro Flughafen die durchschnittliche Distanz aller ausgehenden Routen berechnet. Nutzen Sie die Funktion `std::accumulate`. Speichern Sie die Ergebnisse in `AirportInfo::m_averageRouteLength`.

Aufgabe 3.3: Genetische Programmierung (8 Punkte 1+1+1+2+2+1)

Das Problem des Handlungsreisenden (Traveling Salesman Problem, TSP) beschreibt das Problem der Suche nach der kürzesten Rundreise zwischen einer Menge von Wegpunkten. Mit steigender Anzahl an Wegpunkten wächst die Anzahl an möglichen Pfaden überexponentiell und die exakte Berechnung wird nicht mehr handhabbar.

Zur Lösung der Aufgabe soll deshalb genetische Programmierung genutzt werden, um das TSP anhand einer Entfernungsmatrix (`distance_table`) der 20 größten deutschen Städte zu begutachten. Eine Tour beinhaltet dabei den Besuch einer jeden Stadt. Die Gesamtstrecke einer Tour berechnet sich durch die Summe der Streckenabschnitte inklusive der Rückreisestrecke. Das gesuchte Ergebnis ist eine Tour mit möglichst kurzer Gesamtstrecke.

Nutzen Sie den vorgegeben Programmrahmen in `generic-tsp.cpp`.

a) Implementieren Sie die Funktion `cityDistance`, die zu zwei gegebenen Städten die entsprechende Distanz aus der Distanzmatrix zurückgibt. Machen Sie sich mit dem Befehl `assert` vertraut und geben Sie Zusicherungen an, die für die Wertebereiche und Ungleichheit der übergebenen Parameter gelten.

b) Implementieren Sie die Funktion `tourLength`, die die Länge einer gegebenen Strecke bestimmt und formulieren Sie erneut Assertions zur Zusicherung des gültigen Zustands der Tour. Nutzen Sie dafür die bereits implementierten Funktionen des Programmrahmens.

c) Fügen Sie in `insertCity` die übergebene Stadt an der nächsten freien Position der gegebenen Tour ein. Sichern Sie dabei den Wertebereich der Stadt und die Größe der Tour zu.

d) Der Programmrahmen erzeugt in `generateTours` bereits eine zufällige Anfangspopulation. Implementieren Sie das Kreuzen der bisherigen besten zwei Touren in `crossover`, indem Sie einen Abschnitt aus der einen Tour übernehmen und die restlichen Städte in der Reihenfolge aus der anderen Tour hinzufügen.

e) Erlauben Sie zusätzlich die Möglichkeit, wahrscheinlichkeitsbedingt die Indizes von Städten einer Tour zu vertauschen (Mutation). Implementieren Sie dazu die Funktion `mutate`, die jede Stadt einer Tour mit einer Wahrscheinlichkeit von 2 % mit einer zufälligen anderen Stadt tauscht. Wenden Sie die Mutation in `evolution` an, wobei Sie die dort gerade neu erzeugte Tour und die zwei bisher besten Touren von der Mutation ausnehmen.

f) Visualisieren Sie die Entwicklung der Touren, die als `output.csv` ausgegeben wird, in einem Diagramm und reichen Sie dieses mit Ihrer Abgabe als PDF ein.

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 spätestens **Montag, den 5. Juni um 17:00 Uhr** ein.
- Die Implementierung kann auf einer üblichen Plattform (Windows, Linux, OS X) erfolgen, darf aber keine plattformspezifischen Elemente enthalten, d. h. die Implementierung soll plattformunabhängig entwickelt werden. Lesen Sie bei Fragen zum Kompilieren und Ausführen bitte genau die dem Programmrahmen beiliegende `README.TXT`.

  Bestehen weitere Fragen und Probleme, kontaktieren Sie den Übungsleiter oder nutzen Sie das Forum im Moodle.
- Archivieren Sie zur Abgabe Ihren bearbeiteten Programmrahmen und die PDF-Dateien der schriftlich zu bearbeitenden Aufgaben als Zip-Archiv und ergänzen Sie Ihre Namen im Bezeichner des Zip-Archivs entsprechend im folgenden Format: **`uebung3_vorname1_nachname1_vorname2_nachname2.zip`**. Beachten Sie, dass dabei nur die vollständigen Quelltexte, die CMake-Konfiguration sowie eventuelle Zusatzdaten gepackt werden (alle Dateien, die im gegebenen Programmrahmen vorhanden waren). Laden Sie keine Kompilate und temporären Dateien (*.obj, *.pdb, *.ilk, *.ncb etc.) hoch. Testen Sie vor dem Hochladen, ob die Abgabe fehlerfrei kompiliert und ausgeführt werden kann.
- Reichen Sie Ihr Zip-Archiv im Moodle ein:
  `https://moodle.hpi3d.de/course/view.php?id=96`.