Programmiertechnik II

Sommersemester 2015 – Aufgabenblatt 4

Hasso-Plattner-Institut
Fachgebiet Computergrafische Systeme

Anmerkungen zu Aufgabenblatt 4

Der zu verwendende Programmrahmen steht als Zip-Archiv `uebung4.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 4.1: Algorithmen und Funktionsobjekte in C++ (6 Punkte $^{2+2+2}$)

Die Programmierung von Algorithmen wird in C++ durch Lambda-Ausdrücken 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 `mapreduce.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 Zwischenstop beinhalten. Nutzen Sie 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 4.2: Function Wrapper in C++ (7 Punkte $^{3+3+1}$)

In dieser Aufgabe sollen verschiedene mathematische Funktionen unter Verwendung von Funktionsobjekten (`std::function`) auf eine gegebene Zahlenfolge angewendet werden. Die anzuwendenen Funktionen können dabei über die Kommandozeile spezifiziert werden. Erweitern Sie das Modul `functionwrapping.cpp` wie folgt:

a) Implementieren Sie zunächst die folgenden Funktionen:

- `fibonacci(int x)`: Gibt die Fibonacci-Zahl für `x` zurück.
- `centeredTriangular(int x)`: Gibt die zentrierte Dreieckszahl für `x` zurück.
- `powSum(int x, int n)`: Gibt die Summer der `n`-ten Potenzen für eine Zahlenfolge `1..x` zurück.

b) Über entsprechende Kommandozeilenargumente kann spezifiziert werden, welche der obigen Funktionen auf die Zahlenfolge angewendet werden sollen:

- `-fib`: Anwendung von `fibonacci(int x)`
- `-ct`: Anwendung von `centeredTriangular(int x)`

- `-pow`: Anwendung von `powSum(int x, int n)`

Binden Sie in der Funktion `main()` die ausgewählten Funktionen an jeweils ein Funktionswrapper-Objekt (`std::function`). Fügen Sie die Funktionswrapper-Objekte dem vorgegebenen `functions`-Vector hinzu.

Binden Sie dabei die `fibonacci`-Funktion direkt als Funktionsobjekt, die `centeredTriangular`-Funktion mit Hilfe eines lambda-Ausdrucks und die `powSum`-Funktion mit Hilfe der `std::bind`-Funktion (für den Exponenten soll ein konstanter Wert übergeben werden).

c) Implementieren Sie die Funktion `applyFunctions()`, die alle im `functions`-Vector enthaltenen Funktionen auf eine gegebene Zahl anwendet. Geben Sie die Ergebnisse auf der Kommandozeile aus.

Zusatzaufgabe: Genetische Programmierung (1 Klausurpunkt)

Das Handlungsreisendeproblem (besser bekannt als "Traveling Salesman Problem" oder kurz 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 Berechnung wird somit nicht mehr handhabbar.

Zur Lösung der Aufgabe soll deshalb genetische Programmierung genutzt werden um das TSP anhand einer Entfernungsmatrix 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.

Gehen Sie zur Berechnung wie folgt vor:

- Die Variable `distance_table` beinhaltet die Entfernungen zwischen den Strecken.
- Berechnen Sie eine Menge von zufällig generierten Touren (Anfangspopulation).
- Nutzen Sie das Ergebnis der besten zwei Touren und "kreuzen" Sie diese indem Sie den ersten Abschnitt aus einer und den zweiten Abschnitt aus der anderen Tour vereinen (Kreuzung).
- Erlauben Sie zusätzlich die Möglichkeit wahrscheinlichkeitsbedingt die Indizes von Städten einer Tour zu vertauschen (Mutation).

Nutzen Sie den vorgegeben Programmrahmen in `generic-tsp.cpp` und passen Sie diesen an.

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 **Mittwoch, den 27. Mai um 13:00 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: **`uebung4_matrikNr1_matrikNr2.zip`**. 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`.