Programmieren/Algorithmen & Datenstrukturen 2
- Klausur -

20.07.2016

Zu Beachten:

- Überprüfen Sie Ihr Exemplar auf Vollständigkeit (4 Seiten).
- Tragen Sie Ihren Namen, Vornamen und Matrikelnummer auf dem Login Zettel ein.
- Lassen Sie **alle** ausgegebenen Zettel auf Ihrem Platz!
- **Zulässige Hilfsmittel:** Bücher, installierte C++ Referenz, **Ordner** mit Unterlagen. **Lose Blätter werden von der Aufsicht konfisziert**.
- Lesen Sie zunächst **alle** Aufgaben durch und stellen Sie Verständnisfragen zu Beginn der Klausur.
- **Inhaltliche** Fragen werden während der Klausur **nicht** beantwortet.
- **Bearbeitungszeit:** 180 Minuten.

Punkte:

- Die Klausur ist mit 50 Punkten sicher bestanden.
- Die Punkte entsprechen Prozentpunkten, 100% der Punkte sind damit 100 Punkte.

1 Einleitung

In dieser Klausur werden Sie eine Textdatei einlesen und verarbeiten, die Daten zu **Sternen** enthält. Die Datei enthält dabei folgende Informationen über 247 Sterne:

- Eine **ID** in Form einer eindeutigen Nummer
- Die **Bezeichnung** des Sterns (falls vorhanden) in Form eines Strings, welcher Leerzeichen enthalten kann
- Die **Koordinaten** des Sterns in Form von **drei** Komponenten für X, Y und Z-Koordinate
- Eine Angabe in Form einer Zahl 1 bis 3, welchen Index dieser Stern in einem **Mehrfachsystem** annimmt
- Die ID des **Primärsterns**
- Das **Sternbild** (falls vorhanden), in dem der Stern am Himmel zu sehen ist, in Form einer Abkürzung aus drei Zeichen

Die Sterne sollen mit all Ihren Daten eingelesen werden und abgespeichert werden. Sie sollen zusätzlich Funktionalitäten zum Suchen von Sternen nach bestimmten Kriterien implementieren. Außerdem sollen Sie eine Liste der Mehrfachsysteme erstellen.

2 Regeln

- Schreiben Sie **Kommentare** (nur da) wo benötigt. Beispielsweise wenn Sie sich nicht sicher sind, ob ein bestimmter Algorithmus funktioniert. Oder falls eine oder mehrere Zeilen Ihnen Laufzeit- bzw. Compilerfehler geben, Sie aber sicher sind, diese für eine korrekte Lösung zu benötigen.
- Beachten Sie unbedingt **`const`-correctness** (Bei Nichtbeachtung: **Punktabzug**)
- Etwaig benötigte Konstruktoren, Kopierkonstruktoren, Zuweisungsoperatoren und Destruktoren sind nicht zwangsläufig in den Aufgabenstellungen beschrieben. Fügen Sie diese, **wenn benötigt**, eigenständig hinzu
- Bei der Abgabe erwarte ich ein **lauffähiges** Programm (Bei Nichtbeachtung: **Punktabzug**)
- Einige Aufgaben sind mit dem Hinweis „fakultativ“ gekennzeichnet. Sollten Sie an einer oder mehrerer dieser Teilaufgaben scheitern, so können Sie trotzdem die Klausur weiter bearbeiten. Die in der fakultativen Aufgabe zu erreichenden Punkte erhalten Sie dann natürlich nicht (oder nur teilweise, bei angefangener Lösung).

Hinweis: Sie dürfen zu den Klassen beliebig Methoden und Attribute hinzufügen, wenn Sie für Ihre Lösung einen Bedarf sehen. Dies betrifft vor allem `get`- und `set`-Methoden

3 Aufbau der Textdateien

Ihnen stehen vier verschiedene Textdateien zur Verfügung, die sich lediglich durch ihre **Formatierung** unterscheiden. Schauen Sie am Besten alle Dateien mit einem Texteditor an, und entscheiden Sie anschließend, welche Sie als am „einfachsten“ empfinden.

Es gibt jeweils Versionen, bei denen alle Daten eines Sterns in einer **Zeile** durch Komma getrennt stehen, oder aber jede einzelne Information in einer eigenen Zeile. Weiterhin unterscheiden sich die Versionen darin, ob eine nicht vorhandene Information einfach leer gelassen wurde, oder aber ob dort der String „leer“ steht. Nicht alle Sterne besitzen einen **Namen**, und nicht alle Sterne sind Teil eines **Sternbildes**.

Eine Beispielübersicht (`stars-newline-leer.txt`):

```
0
Sol
0.000005
0.000000
0.000000
leer
1
0
```

(`stars-komma.txt`)

```
0,Sol,0.000005,0.000000,0.000000,,1,0
```

Anmerkungen:

- Die Koordinaten haben ihren Nullpunkt an der Erde
- Mehrfachsysteme bestehen nicht nur aus einer „Sonne“, sondern aus mehreren. In den Daten gibt es zu jedem Stern eine Angabe, welchen Index dieser in einem Mehrfachsystem einnimmt. Die Indizes reichen von 1 bis 3. Weiterhin ist die ID **Hauptsterns** immer angegeben. **Wichtig:** Bei einem Einfachsystem wie dem Sonnensystem sehen Sie, dass Sol den Index 1 besitzt, also der Hauptstern des Systems ist. Die ID des Hauptsterns ist damit ebenfalls 0, die ID von Sol.
  Ein Beispiel eines Mehrfachsystems:

  ```
  17707,,3.433740,5.288370,16.149783,Cam,1,17707
  118255,,3.441942,5.300133,16.192839,Cam,2,17707
  118256,,3.441911,5.300085,16.192691,Cam,3,17707
  ```

  Die drei einzelnen Sterne besitzen unterschiedliche IDs und leicht unterschiedliche Koordinaten. Der Index gibt an, welcher der größte, mittlere bzw. kleinste Stern ist. Der **Hauptstern** aller drei Sterne ist identisch: Es ist die ID des größten Sterns des Systems.

4 Aufgaben

1. Klasse `Stern` Gesamt: 10 Punkte

   Diese Klasse dient zum Sammeln aller Information eines einzelnes Sterns. Im Wesentlichen sind hier nur die entsprechenden Attribute und Methoden zur Ein- und Ausgabe zu implementieren.

   Sie können getter/setter oder andere Methoden später bei Bedarf hinzufügen.

[5 P.] (a) Legen Sie für alle in Aufgabe 1 genannten Information entsprechende Attribute an. Die Datentypen wählen Sie geeignet zu den Daten aus der Textdatei.

[2 P.] (b) Schreiben Sie eine Methode `print()`, welche die wichtigsten Information eines Sterns in etwa so auf der Konsole ausgibt:

```
ID: 84086
Name: Rasalgethi
Koordinaten: -20.9955, -104.71, 27.4009
```

[3 P.] i. **Fakultativ:** Anstatt der Methode `print()` überladen Sie den **Ausgabeoperator** <<.

2. Klasse `Galaxis` Gesamt: 90 Punkte

Die Klasse `Galaxis` dient zur Sammlung aller `Stern`-Objekte und enthält daher eine oder mehrere Datenstrukturen, welche die Objekte speichern.

Hinweis: Alle Teilaufgaben nach der ersten sind nicht zwangsläufig voneinander abhängig und können von Ihnen höchstwahrscheinlich in einer beliebigen Reihenfolge erledigt werden.

Hinweis: Es ist unter Umständen erforderlich, die Sterne in **mehr als einer** Datenstruktur abzulegen. Beispielsweise kann es sinnvoll sein, für verschiedene Teilaufgaben unterschiedliche Strukturen zu verwenden, oder aber die Sterne in zwei gleichen Strukturen gleichzeitig zu halten, allerdings nach unterschiedlichen Kriterien sortiert.

[20 P.] (a) Implementieren Sie einen Konstruktor `Galaxis(const std::string& datei)`, der den als String übergebenen Dateinamen öffnet, und überprüft ob die Datei geöffnet werden konnte.
Wenn die Datei nicht geöffnet werden kann, werfen Sie eine **Exception** des Typs `invalid_argument`.
Lesen Sie den gesamten Inhalt der Datei aus, und speichern Sie alle Sterne in einer geeigneten Datenstruktur ab.

[5 P.] i. **Fakultativ:** Sie implementieren zum Einlesen **eines** Sterns den Streamoperator >>. Dieser liest aus dem Eingabestrom genau einen Stern ein. Der Operator ist für ein Sternobjekt zu implementieren.

[5 P.] (b) Implementieren Sie eine Methode `find(int id)`, die einen Stern mit dieser ID sucht und eine **Referenz** auf diesen zurück liefert.
Ist kein Stern mit dieser ID auffindbar, soll eine **Exception** geworfen werden.

[5 P.] i. Ihre Datenstruktur erlaubt, dass die Suche im Mittel einen Aufwand von $\leq log_2(n)$ hat.

[5 P.] (c) Implementieren Sie eine Methode `find(const std::string& name)`, die einen Stern mit diesem Namen sucht und eine **Referenz** auf diesen zurück liefert.
Ist kein Stern mit diesem Namen auffindbar, soll eine **Exception** geworfen werden.

[5 P.] i. Ihre Datenstruktur erlaubt, dass die Suche im Mittel einen Aufwand von $\leq log_2(n)$ hat.

[10 P.] (d) Berechnen Sie in einer Methode die **Entfernung** zu allen Sternen **mit Namen** und geben Sie diese **sortiert nach Entfernung** auf dem Bildschirm aus.
Sie können die Entfernung relativ einfach aus den Koordinaten berechnen, in dem Sie die Quadrate der einzelnen Komponenten aufaddieren und vom Ergebnis die Wurzel ziehen:

$$dist = \sqrt{X^2 + Y^2 + Z^2} \quad (1)$$

Beispielausgabe:[1]

```
Sol: 5e-06 pc
Proxima Centauri: 1.29576 pc
Rigil Kentaurus: 1.32457 pc
Barnard's Star: 1.82281 pc
Lalande 21185: 2.54615 pc
Sirius: 2.63702 pc
Lacaille 9352: 3.27598 pc
Procyon: 3.51416 pc
Luyten's Star: 3.80276 pc
Kapteyn's Star: 3.9136 pc
Lacaille 8760: 3.94645 pc
Kruger 60: 4.0007 pc
//......usw.
```

[5 P.] i. **Fakultativ:** Sie verwenden für die Sortierung ein **Funktionsobjekt**

[1] Die Abkürzung pc bedeutet „Parsec“, die Einheit die entsteht, wenn Sie die Entfernung wie oben angegeben berechnen.

[15 P.] (e) Erstellen Sie eine Auflistung aller **Dreifachsysteme**, also aller Systeme mit drei Sternen. Geben Sie hierbei lediglich die IDs der entsprechenden Einzelsterne aus.

Beispielausgabe:

```
6442 118069 118070
7358 118074 118075
11538 118147 118148
12673 118166 118167
13608 118179 118180
13735 13738 118182
15805 118217 118218
17707 118255 118256
//.......usw.
```

[7 P.] (f) Es gibt eine weitere Datei `sternbilder.txt`, welche eine Gegenüberstellung der in den Sterninformationen verwendeten **Abkürzungen** der **Sternbilder** und den ausgeschriebenen Bezeichnungen enthält.

Beispiel:

```
Cyg Cygnus
Del Delphinus
Dor Dorado
Dra Draco
```

Lesen Sie im Konstruktor der `Galaxis`-Klasse diese Datei zusätzlich mit ein, und ersetzen die Abkürzungen in allen eingelesenen Sternen durch das ausgeschriebene Pendant.

[8 P.] i. **Fakultativ:** Sie verwenden eine `std::map`, um die Abkürzungen und die ausgeschriebenen Bezeichnungen abzuspeichern, und verwenden anschließend diese map bei der Ersetzung.

Viel Erfolg!