Programmieren/Algorithmen & Datenstrukturen 2
- Klausur -

28.09.2015

Zu Beachten:

- Überprüfen Sie Ihr Exemplar auf Vollständigkeit (3 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

Sie sollen eine Anwendung erstellen, die ein Wörterverzeichnis für Texte erstellt. Dabei werden verschiedene Wörter in Kategorien unterteilt, und es werden für einige Kategorien die Zeilen gespeichert, in der diese Wörter im Text auftreten.
Es sollen folgende **Kategorien** unterschieden werden:

- Nomen (Hauptwörter) beginnen mit einem Großbuchstaben
- Orte sind ebenfalls Nomen, stehen aber zusätzlich in einer getrennten Datei
- Alle übrigen Wörter sind „normale“ Wörter

Der zu untersuchende Text ist ein Auszug aus „Irische Elfenmärchen“ der Gebrüder Grimm. Der Text liegt in der Datei „`elfen.txt`“ vor, die darin vorkommenden Orte sind in der Datei „`places.txt`“ aufgelistet.
Entwickeln Sie ein Programm, welches in der Lage ist, Nomen und Orte sortiert nach Häufigkeit bzw. Alphabet ausgibt.

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 Aufgaben

1. **Klassen `Word` (Wort), `Noun` (Nomen), `Place` (Ort)** Gesamt: 34 Punkte

   Diese drei Klassen stellen die in Abschnitt 1 dargestellten **Kategorien** von Wörtern dar:

   `Word` speichert ein allgemeines Wort aus der Textdatei als `string`

   `Noun` ist ein Wort, welches mit einem Großbuchstaben beginnt. Zu speichern sind hier das Wort als solches (`string`) sowie die Zeilennummer(n), in der das Nomen vorkommt

   `Place` ist ein Nomen, das zusätzlich noch eine Ortsangabe in Form von GPS-Koordinaten (Längen- und Breitengrad) beinhaltet

[12 P.] (a) Implementieren Sie die drei oben genannten Klassen mit den dazugehörigen Attributen!

[16 P.] i. **Fakultativ:** Ihre Klassen sind in einer Vererbungshierarchie, in der `Place` von `Noun` erbt und `Noun` wiederum von `Word`.
Hinweis: Die Vererbung ist nicht unbedingt notwendig, macht aber für die späteren Aufgaben sehr vieles einfacher!

[6 P.] (b) Schreiben Sie für jede Klasse eine Methode

```
1 void <KLASSENNAME>::print(std::ostream& os);
```

welche den Inhalt eines Objektes **sinnvoll** in den als Parameter übergebenen `ostream` schreibt. Vergleiche hierzu die Beispielausgabe am Ende der Aufgabenstellung.

2. **Klasse `Tale` (Sage)** Gesamt: 66 Punkte

   Die Klasse `Tale` dient als Managerklasse für eine Sage. Implementieren Sie das Einlesen der Textdatei(en) innerhalb dieser Klasse.
   Darüber hinaus enthält diese Klasse eine oder mehrere Datenstrukturen zum Abspeichern von `Word`, `Noun` und `Place` Objekten. Ergänzen Sie weitere Attribute und Methoden nach Bedarf.

[8 P.] (a) Lesen Sie im Konstruktor von `Tale` oder in einer gesonderten Methode die Datei „`places.txt`“ ein. Diese ist nach folgendem Schema aufgebaut:

```
1 ORTSNAME laengengrad breitengrad
```

wobei Längen- und Breitengrad als reelle Zahlen angegeben sind. **Hinweis:** Schauen Sie sich die Datei zunächst mit einem Texteditor an!

Lesen Sie **alle** Einträge ein und speichern diese in einem Container, der Attribut der Klasse `Tale` sein soll.

[10 P.] i. **Fakultativ:** Sie verwenden als Container für die Orte eine `std::map`.

[20 P.] (b) Lesen Sie im Konstruktor von `Tale` oder in einer gesonderten Methode die Datei „`elfen.txt`“ ein. Entscheiden Sie dabei für jedes eingelesene Wort, in welche **Kategorie** es passt und speichern Sie es entsprechend ab.
Zur Speicherung sollten Sie eine geeignete Datenstruktur verwenden! Wenn Ihre Datenklassen in einer Vererbungshierarchie sind, denken Sie daran, **Basisklassenzeiger** zu speichern!

Bei der Einteilung der Worte in Kategorien ist folgendes zu beachten:

- Ob es sich bei einem Zeichen um einen Großbuchstaben handelt, können Sie mittels der Funktion `isupper` aus dem Header `cctype` prüfen
- Betrachten Sie **alle** groß geschriebenen Wörter als Nomen, auch wenn das Wort nur deswegen groß geschrieben wird, weil es am Satzanfang steht
- Ein **Ort** ist ein Nomen, das zusätzlich in der Datei `places.txt` steht. Verwenden Sie die gespeicherten Orte aus Punkt a, um ein Wort auf die Kategorie „Ort" zu überprüfen
- Sollten Sie Probleme mit Umlauten haben, so probieren Sie **vor** dem Einlesen folgende Zeile:

```
1 #include <locale>
2 ...
3 setlocale(LC_CTYPE, "de_DE"))
```

Ansonsten ignorieren Sie Fehler mit Umlauten einfach!

[3 P.] i. **Fakultativ:** Die Datei ist ein normaler Text und enthält als solcher **Satzzeichen**. Entfernen Sie aus einem ausgelesenen `string` zunächst **alle** Satzzeichen, bevor Sie das Wort in eine Kategorie einordnen.
Hinweis: Die Headerdatei `cctype` bzw. `ctype.h` enthält hierfür interessante Funktionen, beispielsweise `isAlpha`.

[5 P.] ii. **Fakultativ:** Ermitteln Sie die **Zeilennummer** in der ein Wort steht und speichern Sie zu Nomen und Orten die Zeile, in der diese Wörter vorkommen mit ab.

[20 P.] (c) Erstellen Sie einen **Schlagwortverzeichnis**, indem Sie alle Nomen und Orte **sortiert** auf der Konsole ausgeben. Die Sortierung soll dabei nach folgenden Kriterien arbeiten:

- Ausgabe sortiert nach **Häufigkeit**, öfter auftretende Wörter zuerst
- Innerhalb einer Häufigkeit soll nach **Alphabet** sortiert werden

Der zu bearbeitende Text ergibt folgende Ausgabe:

```
1  Noun: Elfen(3): 1, 17, 20
2  Noun: Haufen(2): 3, 3
3  Noun: Lust(2): 18, 22
4  Noun: Menschen(2): 21, 29
5  Noun: Pferde(2): 29, 31
6  Noun: Sonnenuntergang(2): 14, 17
7  Noun: Abgruenden(1): 5
8  Noun: Aufenthaltsort(1): 3
9  Noun: Auge(1): 8
10 Place: Coirshian(1)@ 56.7599 / -4.31415: 11
11 Noun: Ehrerbietung(1): 26
12 Noun: Einfluss(1): 25
13 Noun: Erdhuegeln(1): 7
14 Noun: Erhoehungen(1): 13
15 Noun: Farbe(1): 16
16 ...<hier abgeschnitten
```

wobei die Zahl in Klammern die Häufigkeit ist, gefolgt von den Zeilennummern.
Hinweis: Ihre Ausgabe muss nicht exakt genau so formatiert sein, sollte aber die gleichen Informationen enthalten!

Viel Erfolg!