1. Belegaufgabe

Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

Prog. mit parametrisierten Datentypen | Sommersemester 2010 | Dozent: Horst Hansen

Ausgabe: 26.4.2010 | Abgabe Gruppe 1: 31.5.2011, Gruppe 2: 7.6.2011

Lernziele:
Mit der Lösung dieser Aufgabe sollen sie zeigen, daß Sie in der Lage sind, Anwendungen unter Verwendung parametrisierter Datentypen zu entwerfen und mit Hilfe der Standard Template Library (STL) von C++ zu implementieren.

Spezifische Ziele:

- Verwendung parametrisierter Datentypen beim Entwurf von Programmen
- Benutzen der Behälterklassen der STL
- Benutzen von Algorithmen aus der STL
- Benutzen der Ein-/Ausgabefunktionen von C++
- Trennen von Anwendungsfunktionalität und Benutzungsschnittstelle (Kommandozeile!)
- Dokumentation von Programmen

Aufgabe: Erstellung und Benutzung eines assoziativen Indexes für Programmtexte

Es soll ein Programm zum Indizieren von Texten erstellt werden. Es wird für jedes Wort ein Index erstellt, der angibt, in welcher Zeile an das Wort auftritt. Der erstellte Index wird in vorgegebener Form (siehe unten) in eine Textdatei und/oder am Terminal ausgegeben. Anschließend kann der Index zum Beantworten von Fragen verwendet werden.

Funktionale Anforderungen

Um Programme mit vergleichbaren Ergebnissen zu erhalten, gelten verbindlich die folgenden Definitionen für die Lösung der Aufgabe:

- Das Programm indiziert die Wörter einer oder mehrerer Eingabedatei(en) und gibt den errechneten Index in die Ausgabedatei und optional am Terminal aus.
- Das Programm verwendet den erstellten Index zum Beantworten von Anfragen des Benutzers.
- Ein Wort ist eine zusammenhängende Folge von Zeichen, die mit einem Unterstrich oder einem Buchstaben beginnt und anschließend Buchstaben, Ziffern, Bindestriche oder Unterstriche enthält.

  Als regulärer Ausdruck: `[A-Za-z_]([A-Za-z0-9]|-|_)*` .
- Alle anderen Zeichen sind *Trennzeichen*, d.h. sie beenden ein Wort.

Anforderungen an die Benutzungsschnittstelle

- Die Steuerung des Programms erfolgt ausschließlich über die Kommandozeile.
- Das Programm soll mit den folgenden Kommandozeilenparametern gestartet werden:
  `<program> <options> <outputfile> <inputfile>*`
  `<program>` : Programmname
  `<options>` :
  - `-p` Ausgabe der Indexliste am Terminal
  - `-i` Erstellen des Indexes
  - `-q=<wort>` : Ausgeben des vollständigen Indexes zum Wort `wort` am Terminal
  - `-s=<wortanfang>` : Ausgeben des vollständigen Indexes zu allen Wörtern mit dem Wortanfang `wortanfang` am Terminal
  - `-t=<dateiname>` : Ausgeben der Indizes zu allen Wörtern, die in der Datei `dateiname` vorkommen, am Terminal
  - Die Anfragen verwenden nicht die Eingabedateien, sondern lesen einen zuvor erstellten Index ein! In diesem Fall ist also `outputfile` die einzulesende Datei und es gibt kein `inputfile`!

  `<outputfile>` : Dateiname der Ausgabedatei mit der Indexliste
  `<inputfile>*` : Liste von Eingabedateien mit zu indizierendem Text
- Ausgabe von aussagekräftigen Meldungen bei fehlerhaften Eingaben auf der Kommandozeile
- Verhindern des Überschreibens von Ausgabedateien (leicht)
- Die Ausgabe des vom Programm erzeugten Wortindexes am Terminal und in die Ausgabedatei soll so aufgebaut sein:
  `<wort> BLANK <dateiindex>+` , wobei
  `<dateiindex> ::= <dateiname> (BLANK <zeilennummer>)*` ist.
  Dabei beginnen Wörter eine neue Zeile, jeder weitere Dateiname beginnt ebenfalls eine neue Zeile, wird aber mindestens ein Zeichen weit eingerückt.
- Die Ausgabe der Wortliste erfolgt stets lexikografisch sortiert.
- Die Ausgabe der Zeilennummern erfolgt in aufsteigender Reihenfolge.

Anforderungen an die Implementierung

- Als Kodierung der Zeichen in den Textdateien und im Index wird *ISO Latin1* verwendet.
- Die programminterne Datenhaltung soll mittels der Behältertypen und der Algorithemn der STL implementiert werden.
- Es dürfen nur Standardbibliotheken von C++ für die Implementierung verwendet werden.
- Die Verwendung globaler Variablen oder von Sprunganweisungen ist und bleibt verboten.
- Die Anwendung wird objektorientiert implementiert: Es gibt also nur Klassen und die globale Funktion `main`.
- Wer noch weitere globale Funktionen benötigt, muß dies schriftlich ausführlich begründen!

- Es wird ein Makefile bereitgestellt für
  - das Erzeugen des ausführbaren Programms
  - das Erzeugen der Programmdokumentation
  - das Löschen aller aus den Programmquellen erzeugten Daten
  - das Überprüfen der Speicherverwendung des Programms mittels des Werkzeugs `valgrind`
- Zum Testen des Programms stehen Testdateien in der Datei `Testdaten.zip` auf der Seite zur Lehrveranstaltung zur Verfügung.

Benotung:
Um eine sehr gute Note erreichen zu können, müssen Sie außer den *Muß-Kriterien* auch alle *Soll-Kriterien* erfüllen. Die Erfüllung einzelner *Soll-Kriterien* führt zu einer schrittweisen Verbesserung der Ausgangsnote *drei* für die Bewertung.
Lösungen, die sich nicht an die unter *Anforderungen an die Implementierung* genannten Verbote halten, werden mit **null** Punkten bzw. der Note 5 (**ungenügend**) bewertet!
Beachten Sie bitte dazu auch die auf der Seite zur Lehrveranstaltung veröffentlichte Notenskala!

Als Lösung sind abzugeben:

- ein **Ausdruck** des Klassendiagramms Ihrer Lösung mit den öffentlichen Methoden
- eine vollständige mit `doxygen` erstellte Dokumentation Ihres Programms (**ohne Ausdruck!!!**)
- ein **Ausdruck** einer schriftlichen Beschreibung der Programmstruktur unter besonderer Berücksichtigung der gewählten Behältertypen und Algorithmen aus der STL
- ein **Ausdruck** des kommentierten Programms (z.B. erstellt mittels `pp`)
- ein **Ausdruck** der Indexinformation zu den Texten `Indextest0.txt` und `Indextest1.txt`

Die Abgabe der Lösung erfolgt durch jede Gruppe von Studierenden (maximal 2 Personen pro Gruppe) persönlich im Rahmen der entsprechenden oben genannten Übungsstunde an den Dozenten. Bei der Abgabe der Lösung an einem Rechner im Übungslabor muß das unter Linux funktionsfähige Programm übersetzt und vorgeführt werden. Beide Studierende einer Gruppe müssen alle Fragen des Dozenten zum Programm beantworten können.

Bewertungskriterien:
Bewertet werden neben der Vorführung mit Erläuterung (siehe oben):

- die korrekte Funktion des Programms
- die objektorientierte Struktur des Programms
- die Robustheit des Programms
- die Lesbarkeit des Programmtextes
- die Beschreibung der Programmstruktur
- die Einhaltung der Programmierrichtlinien
- die Form der schriftlichen Dokumente
- die Extras