1. Belegaufgabe

Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

Parametrisierte Programmierung	Sommersemester 2012	Dozent: Horst Hansen
Ausgabe: 12.4.2012		Abgabe Gruppe 2: 24.5., Gruppe 1: 31.5.

Lernziele:

Mit der Lösung dieser Aufgabe sollen sie zeigen, dass Sie in der Lage sind, unter Verwendung von parametrisierten Datentypen Programme 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 der STL
- Benutzen der Ein-/Ausgabefunktionen von C++
- Trennen von Anwendungsfunktionalität und Benutzungsschnittstelle
- Dokumentation von Programmen

Aufgabe: Programm zum Erstellen eines Wortindexes für Texte und zum Beantworten von Anfragen zum Vorkommen von Wörtern im Text

Schreiben Sie ein Programm, das einen nach *ISO Latin 1* kodierten Text indiziert.

Das Programm soll den Text aus einer Datei lesen und daraus eine Liste von Wörtern (siehe unten) erstellen. Zu jedem Wort soll die Spaltennummer aus dem Text, in der das Wort beginnt, und die Zeilennummer festgehalten werden. Wenn ein Wort mehrfach im Text vorkommt, wird von jedem weiteren Vorkommen auch die Zeilennummer und die Spaltennummer festgehalten.

Diese Liste soll auf dem Terminal ausgegeben oder in eine Datei geschrieben werden. Auf dem erstellten Wortindex sollen Anfragen des Benutzers ausgeführt werden können.

In den folgenden Abschnitten werden die Anforderungen an das zu erstellende Programm aufgeführt. Alle Anforderungen, bei denen nichts anderes angegeben ist, sind *Muß-Kriterien*. Blau gedruckte Anforderungen sind *Soll-Kriterien*.

1 Funktionale Anforderungen

- Das Programm indiziert die Wörter eines Textes und gibt den errechneten Index in eine Ausgabedatei oder am Terminal aus.
- Die Ausgabe des Indexes erfolgt lexikografisch sortiert.
- Das Programm verwendet den erstellten Wortindex zum Beantworten von Benutzeranfragen.

- Ein Wort ist eine zusammenhängende Folge von Zeichen, die mit einem Unterstrich oder Buchstaben beginnt und anschließend Buchstaben, Ziffern oder Unterstriche enthält.

  Als regulärer Ausdruck: [A-Za-z_]([A-Za-z0-9_])* .

- Alle anderen Zeichen sind *Trennzeichen*, d.h. sie beenden ein Wort.

- Zeilennummern und Spaltennummern beginnen jeweils mit 1.

2 Anforderungen an die Benutzungsschnittstelle

- Das Programm soll mit den folgenden Kommandozeilenparametern gestartet werden:

  `<program> <inputfile> <outputfile>`
  `<program>` : Programmname
  `<inputfile>` : Dateiname der Eingabedatei mit dem zu indizierenden Text
  `<outputfile>` : Dateiname der Ausgabedatei für den erstellten Index

- Ausgabe von aussagekräftigen Fehlermeldungen bei fehlerhaften Eingaben auf der Kommandozeile.

- Die zeilenweise Ausgabe des Indexes erfolgt in der Form (sowohl für das Terminal als auch für die Ausgabedatei)

  ```
  <token> (BLANK '('<zeilennummer>','<spaltennummer>')' )+ .
  ```

- Die Ausgabe des Indexes erfolgt lexikografisch sortiert.

- Die Ausgabe der Paare von Zeilennummer und Spaltennummer erfolgt sortiert nach aufsteigenden Zeilennummern, bei gleichen Zeilennummern nach aufsteigenden Spaltennummern.

- Es gibt die folgenden interaktiven Kommandos:

  `w! <Wort>`
  Gib den Index zum Wort `<Wort>` aus.

  `a! <Wortanfang>`
  Gib die Indizes zu allen Wörten aus, die den Wortanfang `<Wortanfang>` besitzen.

  `q!` Beende das Programm.

- Die Dialoge mit dem Benutzer sehen ausschließlich so aus:

  1. Eingabe eines der obigen Kommandos
  2. Antwort des Programms
  3. Weiter mit 1.

3 Allgemeine Anforderungen

- Als Zeichenkodierung der Texte wird *ISO Latin1* verwendet.

- Für die Implementierung dürfen nur die Standardbibliotheken von C und C++ verwendet werden.

- Die programminterne Datenhaltung soll mittels Behältertypen der STL implementiert werden.

- Für die Funktionen des Anwendung sollen möglichst Algorithmen aus der STL verwendet werden.
- Das Programm wird objektorientiert implementiert, d.h. es gibt nur Klassen und als einzige globale Funktion `main`.
- Es gibt keine Klassenmethoden.
- Wie üblich gilt: Die Verwendung von globalen Variablen und der Sprunganweisung `goto` ist verboten.
- Für die Implementierung der Orte des Auftretens von Wörtern ist das Klassentemplate `pair<T,S>` aus der STL zu verwenden.
- Versuchen Sie das Programm so zu implementieren, daß alle Aufgaben in möglichst kurzer Zeit bearbeitet werden.
- Messen Sie mit Hilfe der Funktion `clock` die verbrauchte CPU-Zeit für die Indizierung und jedes Kommando und geben sie sie in der Einheit Sekunden mit drei Nachkommastellen aus.
- Stellen Sie sicher, daß das Programm *vor dem Beenden ohne Fehler* allen dynamisch belegten Arbeitsspeicher wieder freigibt! Zum Überprüfen dieser Eigenschaft verwenden Sie das freie Werkzeug `valgrind`.
- 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 Ihres Programms stehen Testdaten in der Datei `Testdaten.zip` im Verzeichnis `Beleg1` 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 *Allgemeine Anforderungen* 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
- ein **Ausdruck** des kommentierten Quelltextes des Programms (Um Papier zu sparen, können Sie den Quelltext vor dem Drucken mit dem Skript `pp` aufbereiten! Dieses Skript ist auf allen Rechnern unserer Labore installiert.)

- ein **Ausdruck** einer Beschreibung und einer Begründung der von Ihnen im Programm verwendeten Datenstruktur(en)
- ein **Ausdruck** des durch Ihr Programm erstellten Indexes für die Datei `Test.c` .
- **ohne Ausdruck**: Eine mittels `doxygen` erstellte vollständige Programmdokumentation aller Klassen und Methoden

Die Abgabe der Lösung erfolgt durch jede Gruppe von maximal 2 Studierenden persönlich im Rahmen der entsprechenden oben genannten Übungsstunde auf einem Laborrechner an den Dozenten. Bei der Abgabe der Lösung muß jede Gruppe das unter dem Betriebssystem MacOSX funktionsfähige Programm vorführen und erläutern. Jeder Studierende der Gruppe muß in der Lage sein, alle Fragen zur vorgeführten Lösung zu beantworten.

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

- die korrekte Funktion des Programms
- die objektorientierte Struktur
- die Robustheit des Programms
- die Lesbarkeit des Programmtextes
- die Vollständigkeit und Verständlichkeit der Programmdokumentation
- die Form der schriftlichen Dokumente
- die Einhaltung der Programmierrichtlinien
- die Implementierung von Sollkriterien