Bachelorstudiengang „Scientific Programming“ / MATSE Ausbildung

FH Aachen Ausbildungsstandorte Jülich, Köln, Forschungszentrum Jülich
Rechen- und Kommunikationszentrum der RWTH Aachen

“C++”

Prüfung 30.01.2014

Prof. Dr. Alexander Voß, Prof. Dr. Andreas Terstegge

Name	
Vorname	
Matrikelnr.	
Sticknr./Login	
Unterschrift	

	Punkte maximal	**Punkte erreicht**	**Korrigiert durch**
Aufgabe 1	**24**		
Aufgabe 2	**20**		
Aufgabe 3	**20**		
Aufgabe 4	**16**		
Aufgabe 5	**20**		
Gesamtpunkte	**100**		
Note			

Allgemeine Randbedingungen und Anforderungen

- **Es gibt vier elektronische Aufgaben A1, A2, A3, A4 sowie eine schriftliche A5.**
- @Aachen: Sie erhalten auf Ihrem Stick eine zip-Datei mit Grundgerüsten zu den Aufgaben jeweils in Form einer cpp- und einer -hpp Datei, sowie einmal eine test.h Datei, eine ident.h Datei und ein makefile.
  Diese zip-Datei entpacken Sie am besten in einem extra eingerichteten Arbeitsverzeichnis.
- Die cpp-Datei enthält Testcode und die hpp-Datei, in der Regel ein nicht funktionsfähiges, aber compilierbares und lauffähiges Grundgerüst.
  Sie bearbeiten nur die hpp-Dateien und dürfen die cpp-Dateien nicht verändern.
- **Codeblöcke sind je nach Aufgabe durch Einkommentieren der Präprozessormakros in der hpp-Datei zu aktivieren.**
- **In der hpp-Datei geben Sie bitte zur Sicherheit auch Ihren Namen und die Matrikelnummer im Kopf der Datei an.**
- Die test.h dient der Ausgabe der Testresultate (Assert etc.) und das makefile der Erstellung der Programme. Beide Dateien dürfen von Ihnen nicht modifiziert werden.
- **Die ident.h dient der Ausgabe Ihres Namens und der Matrikelnummer im laufenden Programm und muss von Ihnen mit diesen Informationen versehen werden.**
- @Aachen: Sie geben nur eine zip Datei mit dem Namen cpp_<MATRNR>_<NAME> ab, wobei “<MATRNR>” durch Ihre 6-stellige Matrikelnummer und “<NAME>” durch Ihren Namen ohne Umlaute zu ersetzen ist.
  Die abzugebende zip-Datei cpp_<MATRNR>_<NAME> enthält das gesamte Arbeitsverzeichnis.
  Der Befehl zum Erstellen der zip-Datei **im Arbeitsverzeichnis** lautet
  ```
  zip cpp_<MATRNR>_<NAME> *
  ```
  wobei “<MATRNR>” durch Ihre 6-stellige Matrikelnummer und “<NAME>” durch Ihren Namen ohne Umlaute zu ersetzen ist.
- **Beachten Sie: Bei jeder Aufgabe gibt es Abzüge, wenn das Programm nicht compiliert, wenn der Code grob ineffizient, kryptisch oder umständlich bzw. unverständlich ist oder wenn Sie einen erfolgreichen Tipp bekommen haben.**

Aufgabe A1

Zu dieser Aufgabe finden Sie in der Datei A1.hpp zwei vorbereitete Klassen: 'Audioformat' und ‘MP3_Format'. Machen Sie sich mit diesem Code vertraut, und lösen Sie dann folgende Unterpunkte.

a) Schauen Sie sich den Testcode (in A1.cpp) zu dieser Unteraufgabe an, und ändern Sie den Code in der .hpp Datei so, dass der Test funktioniert. Dabei sollen die Rückgabewerte der Methoden in der Klasse Audioformat nicht verändert werden!

b) Erweitern Sie die Klasse MP3_Format so, dass der Testcode für diese Teilaufgabe compiliert und richtig ausgeführt wird!

c) Erweitern Sie die Klasse MP3_Format so, dass der Testcode für diese Teilaufgabe compiliert und richtig ausgeführt wird!
Berücksichtigen Sie dabei auch die globale Variable mp3_counter, die die Anzahl von MP3_Format-Objekten zählen soll.

d) Erweitern Sie die Klasse MP3_Format so, dass der Testcode für diese Teilaufgabe richtig ausgeführt wird!

e) Erweitern Sie die Klasse MP3_Format so, dass der Testcode für diese Teilaufgabe richtig ausgeführt wird!

f) Wandeln Sie die Klasse Audioformat in eine rein abstrakte Klasse um (Dazu gibt es keinen Testfall).

Bewertungsschema			
Aufgabe	**Max. Punkte**	**Erreichte Punkte**	**Kommentar**
a)	**4**		
b)	**4**		
c)	**4**		
d)	**4**		
e)	**4**		
f)	**4**		
Comp.fehler	**-2**		
Ineffizient	**-2**		
Tipp	**-4**		
Gesamt	**24**		

Aufgabe A2

In dieser Aufgabe soll eine Liste von Wörtern, die in einer Datei gespeichert sind, nach einem bestimmen Kriterium sortiert werden.

a) Schreiben Sie die vorbereitet Funktion einlesen(), die die Datei 'Datei.txt' zeilenweise in den vorhandenen vector text einlesen soll.

b) Weisen Sie der Variable lambda1 eine Lambda-Funktion zu, die einen übergebenen char auf einen Vokal (a,e,i,o,u) überprüft. Dabei sollen natürlich Groß- und Kleinbuchstaben erkannt werden. Falls der Parameter ein Vokal ist, wird true zurückgegeben.

c) Weisen Sie der Variable lambda2 eine Lambda-Funktion zu, die die Anzahl der Vokale in einem übergebenen String zählt, und als Rückgabewert zurück liefert. Gleichzeitig soll die globale Variable vokal_counter immer um das jeweilige Ergebnis erhöht werden! (Tipp: std::count_if)

d) Nun sollen zwei Strings bezüglich ihrer Anzahl an Vokalen verglichen werden. Dazu soll die Funktion getComp() mit Leben gefüllt werden. Diese Funktion liefert als Rückgabewert eine Lambda-Funktion, die die beiden übergebenen Strings bzgl. ihrer Anzahl an Vokalen vergleicht. Wenn der erste übergebene String weniger Vokale als der zweite String besitzt, wird true zurückgegeben.

e) In der vorbereiteten Funktion sortieren() soll nun der unter a) eingelesene Vektor text sortiert werden. Dazu können Sie den Rückgabewert der Funktion getComp() aus d) nutzen. (Tipp: std::sort).

Hinweis: Wenn Sie Probleme mit Lambda-Funktionen haben, können Sie mit Punktabzug auch reguläre Funktionen programmieren.

Bewertungsschema			
Aufgabe	**Max. Punkte**	**Erreichte Punkte**	**Kommentar**
a)	**4**		
b)	**4**		
c)	**4**		
d)	**4**		
e)	**4**		
Comp.fehler	**-2**		
Ineffizient	**-2**		
Tipp	**-4**		
Gesamt	**20**		

Aufgabe A3

In dieser Aufgabe geht es darum, eine asynchron laufende Sendefunktion mit Daten, d.h. hier Wörtern, zu versorgen, so dass diese Funktion im Abstand von jeweils 100ms Zeichen "sendet". Nutzen Sie dazu den globalen vector Q (Quelle), um Wörter, also Strings, ablegen zu können und den globalen stringstream S (Senke), um Zeichen zu "senden" (statt Console-Output).

a) Starten Sie in der Funktion start_thread einen Thread, der, solange sich Wörter in diesem Container Q befinden, diese nacheinander und Zeichen für Zeichen sendet. Benutzen Sie die globale Variable T für den Thread.
   Liegt ein Wort in Q an, so entfernt der Thread dieses zunächst aus Q und "sendet" es an S. "Senden" bedeutet, die Zeichen dieses Wortes werden im Abstand von jeweils einer Sekunde an S angefügt.
   Liegt kein Wort an, so sendet der Thread ein "-" an S und wartet 100ms, um erneut zu überprüfen, ob ein Wort vorliegt (um es dann wieder zu senden).

b) Legen Sie in der Funktion send_word ein Wort in Q thread-safe ab. Danach wird in der Testfunktion geprüft, ob alle Wörter sowie Zwischenzeichen "-" gesendet wurden, also in S stehen.

c) Warten Sie in stop_thread auf die Beendigung des Threads. Tipp: Sie brauchen eine Idee/Bedingung, um den Thread zu beenden.

Beachten Sie: Der Zugriff auf globale Variablen ist abzusichern.

Nutzen Sie this_thread::sleep_for, um eine gewisse Zeit zu warten.

Bewertungsschema			
Aufgabe	**Max. Punkte**	**Erreichte Punkte**	**Kommentar**
a)	**12**		
b)	**4**		
c)	**4**		
Comp.fehler	**-2**		
Ineffizient	**-2**		
Tipp	**-4**		
Gesamt	**20**		

Aufgabe A4

In dieser Aufgabe geht es um eine zu schreibende Klasse add_formel, die als Member einen STL-vector von Funktionen der Form int→int enthält, die ausgewertet und aufsummiert werden können.

a) Definieren Sie eine Membervariable v vom Typ STL-vector, die oben genannte Funktionen aufnehmen kann.

b) Schreiben Sie eine Memberfunktion operator+=, der als Argument eine Funktion diesen Typs bekommt und sie dem Vektor v hinzufügt.

c) Schreiben Sie eine Funktion eval, die ein Argument vom Typ int bekommt, die alle in dem Vektor v enthaltene Funktionen damit aufruft und die jeweiligen Rückgabewerte aufsummiert und das Ergebnis zurückgibt.

d) Schreiben Sie einen operator(), der die gleiche Funktionalität wie c) liefert.

Bewertungsschema			
Aufgabe	**Max. Punkte**	**Erreichte Punkte**	**Kommentar**
a)	**4**		
b)	**4**		
c)	**4**		
d)	**4**		
Comp.fehler	**-2**		
Ineffizient	**-2**		
Tipp	**-4**		
Gesamt	**16**		

Aufgabe A5

Beantworten Sie bitte folgende Fragen. Es ist genau eine Antwort korrekt!

a) Templates dienen dazu ...
- ( ) ... abstrakte Klassen zu definieren.
- ( ) ... auftretende Datentypen mit T abkürzen zu können.
- ( ) ... gleichen Code für verschiedene Datentypen zu verwenden.
- ( ) ... einen Methodenaufruf erst zur Laufzeit zu bestimmen.

b) Eine Template-Spezialisierung ...
- ( ) ... ist die Ableitung eines Templates von einem Basis-Template.
- ( ) ... liefert eine andere Code-Basis für spezifizierte Datentypen.
- ( ) ... ist eine spezielle Form von virtuellen Templates.
- ( ) ... wird nur bei der Template-Metaprogrammierung benötigt.

c) Mehrfachvererbung ...
- ( ) ... ist in C++ nicht möglich.
- ( ) ... ist in C++ möglich.
- ( ) ... ist in C++ nur möglich, wenn alle Basisklassen mindestens eine virtuelle Funktion besitzen.
- ( ) ... ist in C++ nur möglich, wenn alle Basisklassen mindestens eine rein-virtuelle Funktion besitzen.

d) Ein lock_guard<> wird verwendet, …
- ( ) ... um beim Auftreten von Ausnahmen ein Freigeben des Mutex sicherzustellen.
- ( ) ... um das kompliziertere Interface eines Mutex zu verbergen.
- ( ) ... um einen Mutex überhaupt erst verwenden zu können.
- ( ) ... um die Synchronisierung mehrerer Mutexe sicherzustellen.

e) Die Move-Semantik ...
- ( ) ... hat im neuen C++11 Standard die bisherigen Copy-Konstruktoren und Zuweisungsoperatoren abgelöst.
- ( ) ... beruht auf der Anwendung von R-Values.
- ( ) ... ist ohne ein Programm, dass Dateien verschiebt, nicht sinnvoll.
- ( ) ... ist ein Verfahren zur Optimierung der Reihenfolge von Berechnugsvorschriften.

Bewertungsschema		
Aufgabe	**Max. Punkte**	**Erreichte Punkte**
a)..e)	**4 je Aufgabe**	