Bachelorstudiengang „Scientific Programming“ / MATSE Ausbildung

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

“C++”

Prüfung 17.09.2013

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

Name	
Vorname	
Matrikelnr.	
Sticknr.	
Unterschrift	

	Punkte maximal	**Punkte erreicht**	**Korrigiert durch**
Aufgabe 1	**50**		
Aufgabe 2	**30**		
Aufgabe 3	**20**		
Gesamtpunkte	**100**		
Note			

Allgemeine Randbedingungen und Anforderungen

- **Es gibt drei Aufgaben A1, A2 und A3.**
- 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 brauchen die cpp-Dateien nicht zu verändern. Testfälle oder Codeblöcke sind ggf. in der hpp-Datei ein- oder auszukommentieren. 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 müssen 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.**
- Sie geben nur eine zip Datei mit dem Namen c_cpp_<MATRNR>_<NAME> ab, wobei “<MATRNR>” durch Ihre 6-stellige Matrikelnummer und “<NAME>” durch Ihren Namen ohne Umlaute ersetzt ist.
  Die abzugebende zip-Datei c_cpp_<MATRNR>_<NAME> enthält mindestens allen Quellcode (*.h, *.hpp, *.c und *.cpp Dateien) und die ident.h Datei, kann aber auch ein zip des gesamten Arbeitsverzeichnisses sein.
- **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

In dieser Aufgabe geht es um einen Datencontainer für ganze Zahlen, genauer eine Menge `set`, die für ein Experiment mit (paarweise verschiedenen) Zufallszahlen benötigt wird. Die interne Repräsentation ist beliebig, nur die Benutzung der std-Klassen map, set, unordered_map und unordered_set ist aus Lizenzrechtlichen Gründen nicht erlaubt. Die Anzahl maximal benötigter Elemente ist im Vorhinein bekannt, so dass der Datencontainer bei der Erzeugung entsprechend dimensioniert werden kann.

Realisieren Sie folgende Anforderungen:

a) Es gibt eine (bereits definierte) Klasse `set` mit einem Konstruktor

```
set(unsigned int max_elements)
```

der ein Objekt dieser Klasse initialisiert. Beispiel: Der Code

```
set s1(5);
```

erzeugt ein leeres Mengenobjekt `s1` der Klasse `set` mit Platz für 5 Zahlen. **[5 P.]**

b) Es gibt einen weiteren, zu definierenden Konstruktor, dem zusätzlich eine beliebige Menge von Zahlen übergeben werden kann. Beispiel: Bei folgender Deklaration des Mengenobjekts `s2` der Klasse `set` ist dieses schon mit den Elementen 1 und 2 initialisiert und bietet insgesamt Platz für 7 Zahlen.

```
set s2(7,{1,2});
```

[5 P.]

c) Die Funktion

```
bool contains(unsigned int n)
```

testet, ob die übergebene Zahl n in der Menge enthalten ist. **[5 P.]**

Die Funktion `test_ctor_contains` testet a), b) und c).

d) Die Funktion

```
void set_value(unsigned int n)
```

fügt die Zahl n in die Menge ein, falls n noch nicht enthalten ist. **[5 P.]**

e) Werden mehr Zahlen in die Menge eingefügt als ursprünglich reservierter Platz vorhanden ist, wird eine Ausnahme geworfen. **[5 P.]**

f) Die Funktion

```
void remove_value(unsigned int n)
```

löscht die Zahl n aus der Menge. **[5 P.]**

Die Funktion `test_set_remove` testet d), e) und f).

g) Die Klasse set enthält einen binären +-Operator, der zwei Mengen zu einer vereinigt. Doppelte Zahlen kommen in der Vereinigungsmenge nur einmal vor. Die maximale Anzahl von Elementen ergibt sich aus der Summe der maximalen Anzahlen beider Operatoren. `test_opplus` testet diese Anforderung. **[5 P.]**

h) Die Klasse `set` enthält einen Ausgabeoperator, so dass die Ausgabe in einen `ostream`, wie in der Funktion `test_output` zu sehen ist, funktioniert. **[5 P.]**

i) Die Implementierung der zuvor genannten Funktionen ist so effizient, dass das Einfügen von 10000 Zufallszahlen aus dem Bereich [0..100000] mindestens doppelt so schnell ist, wie das Einfügen in einen Standardvektor, bei dem bei jedem Einfügen gesucht wird, ob das einzufügende Element schon existiert. Die Funktion `test_speed` misst die Zeiten und testet die Geschwindigkeit. **[5 P.]**

j) Alle Testfälle sind “OK”. **[5 P.]**

Die Testfälle in cpp_A1.cpp rufen die genannten Testfunktionen auf.

Bewertungsschema			
A1	**Mögliche Punkte**	**Erreichte Punkte**	**Kommentar**
a)	**5**		
b)	**5**		
c)	**5**		
d)	**5**		
e)	**5**		
f)	**5**		
g)	**5**		
h)	**5**		
i)	**5**		
j)	**5**		
Compiliert nicht	**-3**		
Ineffizient, Kryptisch	**-2**		
Erfolgreicher Tipp	**-5**		
	50-10		

Aufgabe A2

In dieser Aufgabe geht es um Fehlerbehebung. In cpp_A2.h sind einige Codefragmente, die nicht funktionieren, wie sie sollten. Korrigieren Sie folgende Punkte:

a) In `test_f` soll die Memberfunktion `f` aufgerufen werden, aber das Einkommentieren der Zeile 45 führt zu einem Compilerfehler. Beheben Sie den Fehler in `test_f`, so dass der Aufruf, und somit der Test, funktioniert. **[5 P.]**

b) In `test_g` wird ein Objekt vom Typ `B` angelegt und dessen Funktion `g` aufgerufen – das war zumindest die Intention. Leider wird stattdessen die Funktion `g` aus Klasse `A` ausgeführt, die einen falschen Wert zurückgibt (enum `in_A` statt `in_B`). Modifizieren Sie den Code so, dass `g` aus `B` aufgerufen wird, und der Test funktioniert. **[5 P.]**

c) In `test_g` gibt es ein weiteres Problem, beheben Sie es geeignet. **[5 P.]**

d) In `test_h` wird der Ausgabeoperator der Klasse `B` aufgerufen. Dort soll die Memberfunktion `h` aufgerufen werden – allerdings führt das wiederum zu einem Compilerfehler. Daher ist Zeile 35 auskommentiert. Kommentieren Sie sie wieder ein und ergänzen den Code so, dass der Test funktioniert. **[5 P.]**

e) In `test_n` wird auf den Member `n` der Klasse `B` zugegriffen. Intendiert war allerdings das `n` aus `A`. Korrigieren Sie den Code in Zeile 68 so, dass mit `n` aus der Basisklasse verglichen wird und somit der Test funktioniert. **[5 P.]**

f) In `test_s` soll die Summe über 1/(i*i) von 1 bis n schon während des Compilierens berechnet werden. Dazu wird das Template `sum` rekursiv verwendet und die Summe ergibt sich als initialisierte Membervariable `s`. Leider führt der einkommentierte Aufruf in Zeile 81 zu einem Compilerfehler. Kommentieren Sie die Zeile ein und ergänzen den Code entsprechend, so dass der Aufruf und der Test funktioniert. **[5 P.]**

Die Testfälle in cpp_A2.cpp rufen die genannten Testfunktionen auf.

Bewertungsschema			
A2	**Mögliche Punkte**	**Erreichte Punkte**	**Kommentar**
a)	**5**		
b)	**10**		
c)	**5**		
d)	**5**		
e)	**5**		
f)	**5**		
Compiliert nicht	**-3**		
Erfolgreicher Tipp	**-5**		
	30-10		

Aufgabe A3

In dieser Aufgabe geht es um parallele Summenberechnung, genauer um die Approximation von

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

Da der Grenzwert im Allgemeinen nicht bekannt ist, wird von 1 bis zu einem gegebenen maximalen Index summiert. Realisieren Sie folgende Anforderungen:

a) Die Funktion

```
bool sum(unsigned int number_threads)
```

berechnet die obige Summe von `n0`=1 bis `n1`=1000001 parallel, effizient und thread-safe mit `number_threads` Threads (ohne main-Thread gerechnet). **[15 P.]**

b) Alle Testfälle sind “OK”. **[5 P.]**

Die Testfälle in cpp_A3.cpp rufen die genannte Summenfunktionen mit unterschiedlicher Anzahl von Threads auf.

Bewertungsschema			
A3	**Mögliche Punkte**	**Erreichte Punkte**	**Kommentar**
a)	**15**		
b)	**5**		
Compiliert nicht	**-3**		
Ineffizient, Kryptisch	**-2**		
Erfolgreicher Tipp	**-5**		
	20-10		