

Fabian Hupfeld

[fabian.hupfeld@student.uni-kassel.de](mailto:fabian.hupfeld@student.uni-kassel.de) MaNr.: 35523329

Konstantin Stitz

[uk069836@student.uni-kassel.de](mailto:uk069836@student.uni-kassel.de) MaNr.: 35532242

## Praktikumsaufgaben 1

### Aufgabe 1.)

a.)

`std::map` greift auf den Datentyp `map` der C++ Standardbibliothek zu.

Bei `map` handelt es sich um eine assoziative Containerklasse, welcher die Werte als Key-Value Paare speichert. Hierbei wird mit einem eindeutigen Key (keine gleichen Keys möglich) auf den zugehörigen Wert zugegriffen. Hierbei wird mit einem eindeutigen Key (keine gleichen Keys möglich) auf den zugehörigen Wert zugegriffen.

Die `at()`-Methode ermöglicht einen Zugriff auf die gespeicherten Daten, indem ihr als Parameter der entsprechende Schlüssel übergeben wird. Ist der Schlüssel im `map`-Container vorhanden, so gibt die Funktion eine Referenz auf den gesuchten Wert zurück. Ist der Schlüssel nicht vorhanden, so wird eine `out_of_range`-Exception ausgelöst.

Die `emplace()`-Methode ermöglicht das Einfügen eines neuen Schlüssel-Werte-Paares. Der Funktion wird dabei das entsprechende Schlüssel-Werte-Paar übergeben. Wenn der Schlüssel bereits vorhanden ist, wird das Objekt zwar erstellt, aber sofort wieder gelöscht, ansonsten wird es dann in den `map`-Container eingefügt (sofern keine weitere Exceptions auftreten). Die Rückgabe der Funktion besteht zum einen aus einem Iterator auf das eingefügte Element (oder auf das bereits existierende Element, falls der Schlüssel schon vorhanden war und deshalb das neue Schlüssel-Werte-Paar nicht eingefügt wurde) und zum anderen aus einem Wahrheitswert, der angibt, ob das neue Schlüssel-Werte-Paar eingefügt wurde oder nicht.

Die `erase()`-Methode löscht Elemente aus dem `map`-Container. Dabei gibt es mehrere Möglichkeiten auszuwählen, welche Elemente gelöscht werden soll. Wenn man der Funktion als Parameter einen Iterator auf ein Element übergibt, wird genau dieses Element gelöscht. Man kann auch einen Bereich mit Startpunkt und Endpunkt übergeben, dann werden alle Elemente innerhalb dieses Bereiches gelöscht. Bei der dritten Variante übergibt man der `erase()`-Funktion den Schlüssel des zu löschenden Elements.

b.)

Vektoren sind dynamische Arrays die keine feste Größe haben, sondern ihre Größe mit der Anzahl der Elemente anpassen können.

Der Methode `at()` wird als Parameter die Position des gesuchten Elementes übergeben, liegt diese im Bereich des Arrays, so gibt die Funktion eine Referenz auf das entsprechende Element zurück, andernfalls wird eine „`out_of_range`-Exception“ ausgelöst. Liegt die Position innerhalb des Arrays, kann direkt in konstanter Laufzeit darauf zugegriffen werden, wie es bei einem Array üblich ist.

Die `find()`-Methode auf der `vector`-Datenstruktur sucht Elemente. Dabei werden ihr 3 Parameter übergeben, ein Start und Endwert, welche den Bereich bestimmen in dem gesucht werden soll, und der Wert, nach dem gesucht werden soll. Die Methode gibt einen Iterator auf das Element zurück,

welches in dem übergebenen Bereich als erstes dem gesuchten Wert entspricht. Sie läuft also einfach den entsprechenden Array-Bereich durch, bis entweder der Wert gefunden wurde (also der aktuelle Wert gleich dem der Methode übergebenen Wert ist) oder der Bereich komplett durchlaufen wurde.

#### Aufgabe 2.)

a.)

Durch `const` wird der zurückgegebene Vektor unveränderbar. Das ist möglich, da jeder Eintrag eines Vektors als Teil des Vektors angesehen wird (value-type).

Dadurch kann auch der Inhalt des Vektors nicht verändert werden.

b.)

\* bedeutet, dass der Vektor nicht die einzelnen Books, sondern lediglich die Pointer der Books enthält.

& sorgt dafür, dass nur die Referenz des Vektors zurückgegeben wird, und nicht eine neu erstellt Kopie des Vektors

Zusammenfassend gibt `getBooks()` eine Adresse eines nicht veränderbaren Vektors wieder, welcher Pointer der einzelnen Books enthält.

#### Aufgabe 3.)

a.)

Bei dem Book-Konstruktor in `Book.h` Zeile 8 wurde vergessen einen Standardwert für den bool-Wert `lent` zu vergeben, weshalb ein Aufruf dieses Konstruktors mit 3 Parametern zu einem Fehler führt.

→ default für `lent` auf `false` gesetzt

Bei der Datei `library.cpp` wurde vergessen die Datei `Shelf.h` zu importieren

→ `#include "Shelf.h"`

In der Methode `returnBook` in `Library.cpp` greift `erase()` auf einen `const vector` zu, was nicht möglich ist

→ direkt auf `lendings` zugreifen (siehe zeile 58 `Library.cpp`)

In `Library.cpp` fehlen die Rückgabewerte `true` für den positiven Fall in den Methoden `addShelf` und `addBook`

→ `return true` in Zeile 22 und 31 in `Library.cpp` eingefügt

In der Methode `removeFromLentBooks` in `Visitor.cpp` wurde vergessen abzufragen, ob das Buch auch ausgeliehen wurde oder nicht

→ if Abfrage einfügen, die `false` zurückgibt, wenn das Buch nicht ausgeliehen wurde (Zeilen 37-40 in `Visitor.cpp`)