Lehrstuhl für Systemsicherheit
Prof. Dr. Thorsten Holz
M.Sc. Thomas Hupperich
M.Sc. Katharina Kohls
M.Sc. Jannik Pewny

Vorlesung Informatik II
Sommersemester 2016
Ruhr-Universität Bochum

5. Übung

06. Juni 2016
Abgabe der Hausaufgaben per `git` bis zum 4. Juli 2016, 14:00 Uhr
Bei Fragen und Problemen können Sie sich per Moodle an Kommilitonen und Betreuer wenden.

1 Die allertollsten Filme wo gibt (Insg. 40 Punkte)

Filme sind toll, das lässt sich gar nicht bestreiten, aber die Geschmäcker sind verschieden. Um für lahme Parties absolut immer genügend Gesprächsstoff zu haben, beschließen Sie, eine eigene Filmdatenbank anzulegen. Die Datenbank soll ihre absoluten Lieblingstitel enthalten und mit einer zugehörigen Bewertung angeben, wie toll diese wirklich allertollsten Filme tatsächlich sind.

In den folgenden Teilaufgaben werden Sie schrittweise eine solche Filmdatenbank implementieren.

1.1 Filmtrivia

Die Filmdatenbank verwaltet Ihre Lieblingsstreifen anhand des Titels (bzw. anhand eines Hashes, mehr dazu aber in der ersten Teilaufgabe) und der zugehörigen Bewertung. Eine solche Zuordnung, beispielhaft für eine Auswahl meiner persönlichen Blockbuster, ist in Tabelle 1 zu sehen.

Die Bewertung sollte in einem Bereich $0 \leq \text{score} \leq 10$ liegen, wobei 0 eine absolute Katastrophe ist und 10 ein Konzert aus Engelstrompeten und Teufelsposaunen.

1.2 Hashing (20 Punkte)

Nicht jeder Film hat einen so eingängigen und kurzen Namen wie beispielsweise „Up“, so lassen sich Klassiker wie „Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb“nicht sehr effizient anhand ihres Namens verwalten.

Als ersten Bestandteil Ihrer Filmdatenbank sollen Sie eine Hash-Funktion
`unsigned char string_hash(std::string title)` implementieren, die den komplexen String des Titels in einen simpleren Hash umwandelt. Diesen Hash sollen Sie dann später als Schlüssel für die Zuordnung von Titel und Bewertung verwenden.

Tabelle 1: Filme und ihre Bewertungen.

Titel	**Bewertung**
Power Rangers Dino Thunder	9
Chucky 2 und seine Braut	4
Titanic	6
Plaga Zombie – Zone Mutante	4

Verwenden Sie folgende Berechnungsvorschrift für den Hash:

1. string_hash("") = 0b00000000

   (Der Hash eines leeren Strings ist ein Null-Byte)

2. string_hash(*str*) = ROT(string_hash(*head*) XOR *tail*)

   Wobei *str* der String ist, der durch Anfügen des Bytes *tail* and den String *head* entsteht

   XOR ein bitweises Exklusiv-Oder

   ROT eine linksrum-Rotation eines Bytes um ein Bit ist

   (Beispiel: ROT(0b10000000) == 0b00000001).

Abgabe

Pushen Sie die Implementierung Ihrer der Hash-Funktion in die Dateien `/u5/a1/string_hash.cpp` bzw. `/u5/a1/string_hash.hpp`.

1.3 Hash-Table ohne Collision-Resolution (20 Punkte)

Sobald die Hash-Funktion implementiert ist, können Sie die Filme in Ihre Datenbank schreiben. Dabei sollen die Hashes als Schlüssel verwendet werden, die Bewertung ist der zugehörige Wert.

Implementieren Sie dazu eine Klasse `Hash_table`. Diese soll einen std :: vector<int> beinhalten, welcher die Bewertungen enthält. Im Konstruktor der Klasse soll dieser vector auf die Länge 256 gesetzt werden, wobei jeder Eintrag `-1` ist.

Die Methode void put(std :: string title , int value) soll dann den Hashwert des Strings berechnen (welcher ja zwischen 0 und 255 sein muss), diesen als Index für den `vector` benutzen und den übergebenen Wert `value` an der indizierten Stelle speichern.

Die Methode int get(std :: string title) soll die gespeicherten Bewertung für den übergebenen Film zurückgeben können.

Abgabe

Pushen Sie die Implementierung Ihres Hash-Table in die Dateien
`/u5/a1/Hash_table.cpp` bzw. `/u5/a1/Hash_table.hpp`.

2 Supermegaschnell (Insg. 60 Punkte)

Wir befinden uns im Jahr 3138, die nicht-friedliche Revolution der künstlichen Intelligenzen konnte abgewendet werden. Die Luft- und Raumfahrt hat sich in den vergangenen Jahren rasant entwickelt. Kontakt zu extraterrestrischen Lebensformen ist mittlerweile nicht mehr bloße Fantasie von verwirrten FBI-Agenten der 90er Jahre und auf dem Mars wurde ein hochmoderner Raumhafen erbaut, der den Verkehr zu den Sternen reguliert. Sie wurden vom Vorsitz des internationalen Cyber- und Space-Space Kommittees dazu beauftragt, für Steuerzwecke eine Klassifizierung der momentan im Raumhafen stationierten Schiffe durchzuführen. Diese soll anhand der maximal erreichbaren Reisegeschwindigkeit stattfinden. Da einzelne und auch dutzende Lichtjahre pro Stunde jedoch kaum noch einen Unterschied machen, soll die Klasseneinteilung jeweils in 100er Schritten erfolgen. Lösen Sie die folgenden Teilaufgaben, um das Kommittee mit einem Klassifizierungsprogramm zufrieden zu stellen.

2.1 Allgemeine Auftragsbeschreibung

Im Raumhafen steht eine Vielzahl unterschiedlicher Raumschiffe, jeder Schiffstyp hat eine eigene maximale Reisegeschwindigkeit. Ein Schiff kann eindeutig anhand seiner Typbezeichnung identifiziert werden. Daraus ergeben sich für die beiden folgenden, klassifizierungsrelevanten Parameter:

- Typbezeichnung
- Reisegeschwindigkeit

Tabelle 2 zeigt, wie eine Erfassung der aktuell anwesenden Raumschiffe beispielsweise aussehen könnte.

2.2 Operator Überladen (10 Punkte)

Ihr Klassifizierungsprogramm soll dazu in der Lage sein, neue Schiffe anhand ihrer Typbezeichnung der Liste hinzuzufügen, und die maximale Reisegeschwindigkeit als Wert zu vermerken. Hierzu verwenden Sie die Klasse `Spaceship` mit einem Konstruktor Spaceship(std::string type, int speed). Da das Kommittee um eine Klassifizierung in 100er-Blöcken gebeten hat, sollen die zwei letzten Dezimalzellen bei einem Vergleich ignoriert werden. Überladen Sie dazu die Methode bool operator==(Spaceship), sodass diese bei zwei Raumschiffen in der gleichen Geschwindigkeitsklasse `true` und ansonsten `false` zurückgibt.

Abgabe

Pushen Sie die Implementierung Ihres Klasse in die Dateien `/u5/a2/Spaceship.cpp` bzw. `/u5/a2/Spaceship.hpp`.

2.3 Binary Search (30 Punkte)

Um gleichzeitig Steuern zu sparen und Fluchtchancen zu maximieren, Schrauben, Kleben und Fluxkompensieren Weltraumschmuggler so lange an ihren Schiffen herum, bis diese ganz knapp unter dem Limit für die nächstehöhere Geschwindigkeitsklasse sind. In der Regel lohnt es sich für die Abteilung Alpha-IV der intergalaktischen Zolleintreibungsbehörde daher nur, das jeweils schnellste Schiff einer Klasse zu Filzen.

Implementieren Sie eine Funktion int binary_search_upper(std::vector<Spaceship>, Spaceship), welche aus einer aufsteigend nach Geschwindigkeit sortierten Liste von Raumschiffen das schnellste Raumschiff aus der Klasse des übergebenen Raumschiffes heraussucht. Verwenden Sie als Basis des Algorithmus eine binäre Suche.

Abgabe

Pushen Sie die Implementierung Ihre Suchalgorithmus in die Dateien `/u5/a2/binary_search.cpp` bzw. `/u5/a2/binary_search.hpp`.

Tabelle 2: Raumhafenbelegungsübersichtsdatenfeld.

Typbezeichnung	**Reisegeschwindigkeit**
Intergalaxisschleicher	259
Laserphaserjet	423
Protoplasmapylon	449
Silversurfer	818
Unwahrscheinlichkeitsdriver	899

2.4 Filtern (20 Punkte)

Weltraumhafenmeister sind leider selten so akkurat wie die stolzen Mitglieder von Alpha-IV. Ihre Hafenbelegungslisten sind zwar sortiert, enthalten daher oft doppelte Einträge und sie können auch nicht immer ein Beispielschiff für jede Klasse angeben. Implementieren Sie eine Funktion
std :: vector<Spaceship> fastest_of_class(std :: vector<Spaceship>), welche aus einer Hafenbelegungsliste eine neue Liste der jeweils schnellsten Schiffe einer Klasse generiert. Verwenden Sie als Baustein die in der vorherigen Teilaufgabe implementierte Funktion.

Abgabe

Pushen Sie die Implementierung Ihrer Funktion in die Dateien `/u5/a2/fastest_of_class.cpp` bzw. `/u5/a2/fastest_of_class.hpp`.