Lehrstuhl für Algorith.
und Datenstrukturen
Prof. Dr. Hannah Bast
Axel Lehmann

Programmieren in C++
SS 2018

http://ad-wiki.informatik.uni-freiburg.de/teaching

UNI FREIBURG

Übungsblatt 8

Abgabe bis Dienstag, den **26. Juni 2018** um **12:00 Uhr**

Aufgabe 1 (10 Punkte)

Deklarieren und implementieren Sie eine Klasse *Set<T>* (in den Dateien *Set.h* und *Set.cpp*), die eine Menge von Objekten vom Typ T verwaltet und die folgenden Operationen unterstützt:

insert: Einfügen eines gegebenen Objektes vom Typ T, sofern es noch nicht in der Menge ist (sonst soll die Operation nichts tun)

erase: Löschen eines gegebenen Objektes von Typ T, sofern es in der Menge ist (sonst soll die Operation nichts tun)

lookup: Nachschauen eines gegebenem Objektes vom Typ T: falls es in der Menge ist, gebe *true* zurück, sonst *false*

Die Methoden, die die drei genannten Operationen realisieren, sollten *public* sein (ebenso der Konstruktor und der Destruktor), alles andere *private*. Achten Sie wieder auf const-correctness und eine sinnvolle Übergabe der Argumente, bei der nicht unnötig kopiert wird, falls der Objekttyp sehr groß ist.

Alle drei Operationen sollten in Zeit $O(n)$ laufen, wenn n die Anzahl der Elemente ist, die aktuell in der Menge gespeichert sind. Überlegen Sie selber, welche (einfache) Datenstruktur dafür geeignet ist. Wenn Sie viel mehr als 10 Zeilen Code pro Methode schreiben, denken Sie zu kompliziert.

Auf dem Wiki sind wieder Tests vorgegeben (in einer Datei *SetTest.cpp*). Diese sollten wie gehabt mit Ihrem Code zusammen kompilieren und fehlerfrei durchlaufen. Auch *valgrind* sollte fehlerfrei durchlaufen. Sie brauchen für dieses Übungsblatt kein *...Main* Programm.

[keine Sorge, es kommt noch was]

Aufgabe 2 (10 Punkte)

Deklarieren und implementieren Sie eine Spezialisierung der Klasse *Set* aus Aufgabe 1 für den Typ *char* (in denselben Dateien *Set.h* und *Set.cpp* wie Aufgabe 1). Es sollen ebenfalls die Operationen *insert*, *erase* und *lookup* zur Verfügung stehen, mit der gleichen Funktionalität wie in Aufgabe 1 beschrieben. Die Laufzeit soll jetzt allerdings für alle drei Operationen $O(1)$ sein. Das geht auch mit einer relativ einfachen Datenstruktur, und zwar weil *char* so ein "kleiner" Typ ist.

Die anderen Anforderungen von Aufgabe 1 gelten sinngemäß. Insbesondere sind auch für diese Aufgabe die Tests vorgegeben (in einer Datei *SetCharTest.cpp*, siehe Wiki), die zusammen mit Ihrem Code kompilieren und fehlerfrei durchlaufen sollten.

Optional: Implementieren Sie Ihre Spezialisierung möglichst speichereffizient mit Hilfe der Bit-Operationen, die Sie in der Vorlesung gelernt haben (es geht mit 32 Bytes für ein *Set<char>* Objekt). Verwenden Sie dazu *uint64_t*, dafür braucht man *#include <stdint.h>*. Zum Bitschieben müssen Sie dann so etwas schreiben wie *uint64_t(1) << 63*, sonst bekommen Sie Probleme. Dieser Teil ist wohlgemerkt freiwillig, man kann die volle Punktzahl auch ohne eine speichereffiziente Implementierung erreichen. Es ist allerdings ganz nett und nicht viel mehr Code.

Laden Sie wie gehabt alle Code-Dateien und das Makefile in unser SVN hoch, in einem neuen Unterverzeichnis *blatt-08*. Es gelten weiterhin die 10 Gebote von der letzten Seite des Ü1 und nehmen Sie die Ratschläge Ihres Tutors / Ihrer Tutorin ernst.

Laden Sie wie gehabt auch eine Datei *erfahrungen.txt* hoch (im Unterordner *blatt-08*), in der sie kurz Ihre Erfahrungen mit dem Ü8 und der Vorlesung dazu beschreiben.

Mit welchem Ergebnis wird Deutschland am 27. Juni 2018 vorzeitig aus der WM ausscheiden und was machen Sie dann mit der freigewordenen Zeit?