Professur für Algorithmen
und Datenstrukturen
Prof. Dr. Hannah Bast
Johannes Kalmbach

Programmieren in C++
SS 2022

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

UNI FREIBURG

Übungsblatt 8

Abgabe bis Dienstag, den **28. Juni 2022** um **12:00 Uhr**

Aufgabe 1 (10 Punkte)

Deklarieren und implementieren Sie eine templatisierte 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 enthalten ist (wenn das Element bereits enthalten ist, soll die Operation nichts tun).

erase: Löschen eines gegebenen Objektes vom Typ T, sofern es in der Menge enthalten ist (wenn das Element nicht enthalten ist, soll die Operation nichts tun).

lookup: Überprüfen, ob ein gegebenes Objekt vom Typ T in der Menge enthalten ist (wenn das Element enthalten ist, soll die Operation *true* zurückgeben, ansonsten *false*).

Die Methoden, die die drei genannten Operationen realisieren, sowie der Konstruktor und der Destruktor sollen *public* sein, 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 sollen in Zeit $O(n)$ laufen, wenn n die Anzahl der Elemente ist, die aktuell in der Menge gespeichert sind. Überlegen Sie selbst, 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 sollen wie gehabt mit Ihrem Code zusammen kompilieren und fehlerfrei durchlaufen. Achten Sie auch wieder wie gehabt darauf, dass *valgrind –leak-check=full* fehlerfrei durchläuft. Für dieses Übungsblatt müssen Sie kein *...Main* Programm schreiben. Bei den Test finden Sie auch eine aktualisierte Version des *Makefile*s, welches mehrere **Test.cpp*-Dateien unterstützt.

[mit Subtyp BA.5 umblättern]

Aufgabe 2 (10 Punkte)

Deklarieren und implementieren Sie eine Spezialisierung der Klasse *Set* aus Aufgabe 1 für den Typ *unsigned 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 konstant sein (das heißt, nicht davon abhängen, wie wie viele Elemente in der Menge gespeichert sind). Die Laufzeit sollte auch deutlich schneller sein, als wenn man die allgemeine Implementierung aus Aufgabe 1 benutzen würde (Sie müssen die Laufzeit nicht messen, ein gutes Argument genügt). Das geht mit einer relativ einfachen Datenstruktur, und zwar weil *unsigned 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 kennengelernt haben. Und zwar sollte die Größe von einem *Set<unsigned char>* Objekt nur 32 Bytes sein. Verwenden Sie dazu *uint64_t*, dafür braucht man *#include <cstdint>*. Zum Bitschieben müssen Sie dann so etwas schreiben wie *uint64_t(1) << 63*, sonst meckert der Compiler. 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 und nehmen Sie die Ratschläge Ihres Tutors 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.