UNIVERSITÄT BONN

Algorithmen und Programmierung
Übungsblatt 12
WS 2022/23

Dr. Felix Jonathan Boes
Benedikt Bastin, Ellen Bundschuh, Anna Höpfner, Gina Muuss, Adrian Oeyen, Felix Roth, Thore Wolf

Ausgabe: 09.01.2023
Abgabe: keine Abgabe; Präsentation in der Übung

Aufgabe 1 (Geordnete Abbildungen und Geordnete Mengen)**.** In dieser Aufgabe weisen Sie nach, dass Sie sich an den Umgang mit den Standardtypen für geordneten Mengen (`std::set`) und geordnete Abbildungen (`std::map`) gewöhnt haben. Legen Sie in Ihrem Abgabe-Repository einen neuen Branch mit dem Namen `Mensa` an. Verwenden Sie dort die übliche Projektstruktur.

(1) Erweitern Sie Ihre Klasse `Studi` um die konstante Membervariable `Matrikelnummer`.

(2) Durch die Matrikelnummer sind `Studis` intrinsisch miteinander vergleichbar. Erweitern Sie Ihre Klasse `Studi` um die folgenden Operationen.

```
class Studi {
public:
  /* Weitere Memberfunktionen und Membervariablen */
  bool operator<(const Studi& other) const;
  bool operator>(const Studi& other) const;
  bool operator==(const Studi& other) const;
  bool operator!=(const Studi& other) const;
  /* Weitere Memberfunktionen und Membervariablen */
};
```

(3) Den Klassen `std::set` und `std::map` muss der Typ eines Funktionsobjekts mitgegeben werden, welches Studis miteinander vergleicht. Um die natürliche Ordnung der Studis zu kommunizieren, können Sie folgendes Funktionsobjekt verwenden.

```
class StudiCmp {
public:
  bool operator() (const Studi& x, const Studi& y) const {
    return x<y;
  }
};
```

(4) Machen Sie sich mit den Memberfunktionen der Klasse `std::set<Studi, StudiCmp>` vertraut[1].

(5) Legen Sie (zu Demonstrationszwecken) eine Menge mit drei Studis an.

(6) Nutzen Sie `typedef` um für den Typ `std::set<Studi, StudiCmp>` den Alternativnamen `Studimenge` einzuführen.

[1] Siehe `https://en.cppreference.com/w/cpp/container/set`

(7) Machen Sie sich mit den Memberfunktionen von `std::map<Studi, Studimenge, StudiCmp >` vertraut[2] und führen sie für den Typ `std::map<Studi, Studimenge, StudiCmp>` den Alternativnamen `Mensaabbildung` ein.

(8) Legen Sie (zu Demonstrationszwecken) eine Mensaabbildung an. Diese soll verwendet werden um zu modellieren, mit welcher Menge von Studis ein gegebener Studi gern in die Mensa gehen würde. Modellieren Sie mit dieser Abbildung, dass Alex gern mit Tobi, Pascal und Lisa in die Mensa geht; Frauke gern mit Tobi, Alex und Fatma in die Mensa geht und Lisa gern mit Frauke und Alex in die Mensa geht. Drucken Sie diese Beziehung (nur unter Verwendung der Mensaabbildung, in einem Range-Based-For-Loop) auf der Standardausgabe nachvollziehbar aus. Dabei dürfen Sie ausnutzen, dass über die Keys in aufsteigender Reihenfolge iteriert wird.

Aufgabe 2 (Gerichtete Graphen wie in der Vorlesung)**.** In Ihrem Abgabe-Repository hat Flavius einen Branch mit dem Namen `Zettel_12` angelegt. Erweitern Sie die dort, unter Verwendung der Projektstruktur, die in der Vorlesung vorgestellt Klasse `GerichteterGraph` indem Sie die Memberfunktionen zur Breitensuche oder Tiefensuche implementieren und Ihre Implementierung demonstrieren. Um das Projekt zu kompilieren, gehen Sie wie üblich vor.

Aufgabe 3 (Gewichteter, gerichteter Graph mit reiner Typerweiterung)**.** In dieser Aufgabe modellieren Sie gerichtete Graphen als Subtyp durch reine Typerweiterung. Lösen Sie folgende Teilaufgaben.

(1) Gehen Sie von Ihrem UML Diagram aus Aufgabe 11.2 aus und modellieren Sie den Typ GewichteterGerichteterGraph durch reine Typerweiterung (insbesondere sollen Sie die bereits erhaltenen Kanten verwenden und keine Memberfunktion überschreiben).

(2) Implementieren Sie die zugehörige Klasse die durch öffentliche Subklassenbildung entsteht und demonstrieren Sie Ihre Implementierung (unter Verwendung der Projektstruktur).

(3) *Bonus*: Identifizieren Sie wesentliche Probleme, welche die reine Typerweiterung hier mitbringt.

[2] Siehe `https://en.cppreference.com/w/cpp/container/map`