SWE3 HÜ 4: Templates & Concepts

Erik Pitzer <erik.pitzer@fh-hagenberg.at>

Medizin- und Bio-Informatik – WS 2024/25

Name

Aufwand (in h)

Punkte

Aufgabe	Lösungsidee	Implementierung	Tests	Gesamtpunkte
1	20% / 20P	50% / 50P	30% / 30P	100

Aufgabe 1: Klasse `rational_t` erweitern

Erweitern Sie Ihre Klasse `rational_t` vom vorhergehenden Übungszettel um die folgende Funktionalitäten:

1. Implementieren Sie eine Methode `inverse`, die eine rationale Zahl durch ihren Kehrwert ersetzt.
2. Parametrieren Sie Ihre Klasse und wandeln Sie sie zu einem generischen Datentyp `rational_t<T>` um, `T` ist dabei jener Datentyp, von dem Zähler und Nenner sind.
3. Ihre Klasse `rational_t<T>` exportiert den Datentyp `T` als `value_type`.
4. Implementieren Sie Ihre Klasse `rational_t<T>` so, dass man damit nicht nur rationale Zahlen über $\mathbb{Z}$ sondern über beliebige Bereiche bilden kann. Implementieren Sie zu Testzwecken einen Datentyp `matrix_t<T>` und bilden Sie damit rationale Zahlen vom Typ `rational_t<matrix_t<T>>`, also Bruchzahlen über Matrizen über `T`. **Hinweis:** Es reicht, wenn Sie den Datentyp `matrix_t<T>` als Matrix der Größe 1×1 ausführen.
5. Überlegen Sie, welche Operationen der Datentyp `T` unterstützen muss, damit ihn Ihre Klasse `rational_t<T>` verwenden kann. Schreiben Sie die Klasse `rational_t<T>` so, dass sie möglichst wenig Anforderungen an den Datentyp `T` stellt. (Listen Sie diese Anforderungen auch explizit in der Dokumentation auf.)
   a) Erstellen Sie *entweder* eine Datei `operations.h`, die im Namensraum `ops` die folgenden Funktionen implementiert:
      - `T abs (T const & a);`
      - `bool divides (T const & a, T const & b);`
      - `bool equals (T const & a, T const & b);`
      - `T gcd (T a, T b);`
      - `bool is_negative (T const & a);`
      - `bool is_zero (T const & a);`
      - `T negate (T const & a);`
      - `T remainder (T const & a, T const & b);`

   b) *Alternativ* dazu entwickeln Sie ein `concept rational` das diese Anforderungen beschreibt.
6. Obige Funktionen sind generisch (Typvariable `T`) zu implementieren sowie `inline` auszuführen. Überladen Sie außerdem alle Funktionen für den Datentyp `int`. Ihre Klasse `rational_t<T>` verwendet natürlich alle angegebenen Funktionen.
7. Die Klassen `rational_t<T>` und `matrix_t<T>` implementieren ihre Operatoren `inline` als zweistellige friend-Funktionen („Barton-Nackman Trick“).

Bitte beachten Sie: Testfälle sind ein Musskriterium bei der Punktevergabe. Enthält eine Ausarbeitung keine Testfälle, so werden dafür auch keine Punkte vergeben. Testfälle sind auf die entsprechenden Teilaufgaben zu beziehen. Es muss aus der Testfallausgabe klar ersichtlich sein, auf welche Teilaufgabe sich ein Testfall bezieht.

Die Testfälle sind entsprechend den Teilaufgaben laut Angabe zu reihen. Ein Testfall schreibt die folgenden Informationen aus: Testfallname, was wird getestet (Bezug zur Angabe), erwarteter Output, tatsächlicher Output, Test erfolgreich/nicht erfolgreich. Ist ein Test nicht erfolgreich, so kann eine Beschreibung der vermuteten Fehlerursache bzw. der durchgeführten Fehlersuche doch noch Punkte bringen.

Falls Sie Google-Test einsetzen, erstellen Sie bitte nur ein Testprojekt in dem alle Beispiele getestet werden.

Anmerkungen: (1) Geben Sie für Ihre Problemlösungen auch Lösungsideen an. (2) Kommentieren Sie Ihre Algorithmen ausführlich. (3) Strukturieren Sie Ihre Programme sauber. (4) Geben Sie ausreichend Testfälle ab und prüfen Sie alle Eingabedaten auf ihre Gültigkeit.