Babeș-Bolyai Universität, Cluj Napoca
Fakultät für Mathematik und Informatik
Objekt-Orientierte Programmierung, 2021

Labor 2

Diskussion: 15./16./17. März, **Abgabe:** 21. März, **Punkte:** 4/50

Schreiben Sie eine C++ Konsolenanwendung, welche **eine** der beiden Aufgaben A oder B löst. In jeder Datei, für jede Klasse und jede Funktion ist es **erforderlich** aussagekräftige Kommentare zu schreiben!

Die Lösung besteht aus 3 Dateien mit dem `PREFIX`: **`L2_Nachname_Vorname_AufgabeX`** (wobei X der Buchstabe der gewählten Aufgabe).
Das Programm muss mit dem GnuCompiler compilierbar sein: **`g++ -std=c++17`**

Aufgabe A

Entwickeln Sie eine Klasse *Rational* zur Darstellung und Verarbeitung rationaler Zahlenwerte.

Folgende Dateien sind im Projekt zu erzeugen:

- `PREFIX_rational.h` enthält die Definition der Klasse
- `PREFIX_rational.cpp` enthält die Implementierung
- `PREFIX_rational_test.cpp` enthält `main()` mit den geforderten Testroutinen und Funktionen

Der Zähler (numerator) - und Nenner (denominator) sind als double dargestellt, auf die von außen kein Zugriff erlaubt ist. Zur Anzeige der aktuellen Werte eines Objektes dieser Klasse sollen die Zugriffsmethoden realisiert sein. Die arithmetischen Grundoperationen `+,*,/` für die Verknüpfung von rationalen Zahlenwerten sollen als Methoden realisiert werden.

1. Schreiben Sie die Klasse ***Rational*** mit den erforderlichen Datenkomponenten, dem spezifizierten Konstruktor sowie den Zugriffsmethoden `get_numerator`, `get_denominator`.
2. Erstellen Sie die Methoden `add`, `substract`, `multiply`, und `divide` so, dass die Datenkomponenten des Objektes unverändert bleiben, der zweite Operand als Parameter übergeben wird und der Rückgabewert das Ergebnis der Operation enthält. Testen Sie diese Operationen!
3. Erstellen Sie eine `reduce` Methode, welche aus dem Zähler und dem Nenner den größten gemeinsamen Teiler entfernt. Testen Sie diese Operation!
4. Erstellen Sie eine `inverse` Methode, welche eine rationale Zahl umkehrt. Testen Sie diese Operation!
5. Erstellen Sie eine `compare` Methode, welche zwei rationale Zahlen vergleicht. Testen Sie diese Operation!
6. Erstellen Sie (in `main`) eine Reihe der Objekte vom Typ Rational.
7. Berechnen Sie (in `main`) die Gesamtsumme der Reihe.

Bonus

Ermögliche die übliche infix-Schreibweise für die Operationen +, -, *, /, anstatt der Methoden

Labor 2

Diskussion: 15./16./17. März, **Abgabe:** 21. März, **Punkte:** 4/50

Aufgabe B

Entwickeln Sie eine Klasse *Complex* zur Darstellung und Verarbeitung komplexer Zahlenwerte. Folgende Dateien sind im Projekt zu erzeugen:

- `PREFIX_complex.h` enthält die Definition der Klasse
- `PREFIX_complex.cpp` enthält die Implementierung
- `PREFIX_complex_test.cpp` enthält `main()` mit den geforderten Testroutinen und Funktionen

Der Real- und Imaginärteil sind als double dargestellt, auf die von außen kein Zugriff erlaubt ist. Zur Anzeige der aktuellen Werte eines Objektes dieser Klasse sollen die Zugriffsmethoden realisiert sein. Die arithmetischen Grundoperationen `+,*,/` für die Verknüpfung von komplexen Zahlenwerten sollen als Methoden realisiert werden.

1. Umsetzen Sie die Klasse *Complex* mit den erforderlichen Datenkomponenten, dem spezifizierten Konstruktor sowie den Zugriffsmethoden `get_imaginary`, `get_real`.
2. Erstellen Sie die Methoden `add`, `substract`, `multiply`, und `divide` so, dass die Datenkomponenten des Objektes unverändert bleiben der zweite Operand als Parameter übergeben wird und der Rückgabewert das Ergebnis der Operation enthält. Testen Sie diese Operationen!
3. Erstellen Sie eine `absolute` Methode, welche den Betrag einer komplexen Zahl berechnet. Testen Sie diese Operation!
4. Erstellen Sie eine `compute_polar` Methode, welche eine komplexe Zahl in die Polarfom transformiert. Testen Sie diese Operation!
5. Erstellen Sie eine `text` Methode, welche eine komplexe Zahl als Zeichenkette darstellt. Testen Sie diese Operation!
6. Erstellen Sie (in `main`) eine Reihe der Objekte vom Typ Complex.
7. Berechnen Sie (in `main`) die Gesamtsumme der Reihe.

Bonus

Ermögliche die übliche infix-Schreibweise für die Operationen +, -, *, /, anstatt der Methoden.