Babeș-Bolyai Universität, Cluj Napoca
Fakultät für Mathematik und Informatik
Objekt-Orientierte Programmierung, 2022

Labor 2

Diskussion: Kalenderw. 12, **Abgabe:** Kalenderw. 13, **Punkte:** 4/55

Schreibe eine C++ Konsolenanwendung, das 1 Problem aus der untenstehenden List löst. Wähle das Problem entsprechend der letzten Ziffer Deiner *Matrikelnummer MODULO 3*.

Anforderungen:

1) Implementierung einer Klasse und geeignete Tests (**siehe Vorlesung 3**)
2) Die Lösung besteht aus 3 Dateien mit dem PREFIX: **`L2_Nachname_Vorname_ProblemX`** (wobei X die Zahl der gewählten Aufgabe).
3) Schreibe eine Skriptdatei (.bat oder .sh) mit den Befehlen mit denen das Programm kompiliert wird (mit GnuCompiler: **`g++ -std=c++20`**) und ausgeführt wird.

Bewertungskriterien:

1) Der Programmcode muss fehlerfrei kompilierbar sein.
2) Das Programm muss ausführbar sein und das korrekte Ergebnis liefern.
3) Du musst den Code erklären können.
4) Der Code muss gut lesbar sein.

Problem 0

Entwickeln Sie eine Klasse zur Darstellung und Verarbeitung von **metrischen Längenangaben**. Folgende Dateien sind im Projekt zu erzeugen:

- `PREFIX_length.h` enthält die Definition der Klasse
- `PREFIX_length.cpp` enthält die Implementierung
- `PREFIX_length_test.cpp` enthält `main()` mit **Testroutinen für jede Methode**

Ein Objekt der Klasse Länge sollte sowohl eine Fliesskommazahl für die Größe als auch einen String für Einheit beinhalten. Auf diese Attribute sollte außen kein Zugriff erlaubt sein. Außerdem sollten alle Methoden sicherstellen, dass nur kompatible Längeneinheiten miteinander verarbeitet werden (z.B. 5m + 3,7m = 8,8m oder 9,3km+73,4km=82,7km)

1. Schreiben Sie die Klasse ***Length*** mit den erforderlichen Datenkomponenten, einem spezifizierten Konstruktor sowie den Zugriffsmethoden `get_value`, `get_unit`.
2. Erstellen Sie die Methoden `add`, `substract` so, dass die Datenkomponenten des Objektes unverändert bleiben, der zweite Operand als Parameter übergeben wird und der Rückgabewert das Ergebnis der Operation enthält.
3. Erstellen Sie eine `scale` Methode, welche eine Länge mit einer Zahl multipliziert.
4. Erstellen Sie eine `divide` Methode, welche eine Länge durch eine Zahl dividiert.
5. Erstellen Sie eine `text` Methode, welche den Länge mit Einheit als Zeichenkette darstellt.
6. Erstellen Sie eine `compare` Methode, welche zwei Längen vergleicht, wobei len1.compare(len2) folgende Rückgabewerte hat:

   -1 für len1<len2, 0 len1=len2, +1 für len1>len2
7. Das Programm soll vom Benutzer 2 Längengaben und 1 Zahl einlesen, und damit die Funktionen der Addition, Subtraktion, Skalierung und Vergleich demonstrieren.

Labor 2

Diskussion: Kalenderw. 12, **Abgabe:** Kalenderw. 13, **Punkte:** 4/55

Bonuspunkt (+1)

Ermögliche die Konvertierung von Längen mit einer Methode und einer internen Konversionstabelle (z.B. km -> mi, cm -> in)

Bonuspunkt (+1)

Ermögliche die übliche infix-Schreibweise für die Operationen +, -, *, /, zusätzlich zu den Methoden.

Problem 1

Entwickeln Sie eine Klasse zur Darstellung und Verarbeitung von **Geldbeträgen**. Folgende Dateien sind im Projekt zu erzeugen:

- `PREFIX_money.h` enthält die Definition der Klasse
- `PREFIX_money.cpp` enthält die Implementierung
- `PREFIX_money_test.cpp` enthält `main()` mit **Testroutinen für jede Methode**

Objekte dieser Klasse sollten sehr grosse Beträge mit genau 2 Nachkommastellen verarbeiten können, und auch die Währung (z.B. USD, EUR) beinhalten. Auf diese Attribute sollte außen kein Zugriff erlaubt sein. Die Methoden sollten die Verarbeitung von Beträgen nur derselben Währung zulassen.

1. Umsetzen Sie die Klasse ***Money*** mit den erforderlichen Datenkomponenten, dem spezifizierten Konstruktor sowie den Zugriffsmethoden `get_value`, `get_currency`.
2. Erstellen Sie die Methoden `add`, `substract` so, dass die Datenkomponenten des Objektes unverändert bleiben, der zweite Operand als Parameter übergeben wird und der Rückgabewert das Ergebnis der Operation enthält.
3. Erstellen Sie eine `scale` Methode, welche einen Geldbetrag mit einer Zahl multipliziert.
4. Erstellen Sie eine `divide` Methode, welche eine Geldbetrag durch eine Zahl dividiert.
5. Erstellen Sie eine `text` Methode, welche den Geldbetrag mit Währung als Zeichenkette darstellt.
6. Erstellen Sie eine `compare` Methode, welche zwei Geldbeträge vergleicht, wobei amount1.compare(amount2) folgende Rückgabewerte hat:
   -1 für amount 1< amount 2, 0 für amount1=amount 2, +1 für amount 1> amount2
7. Das Programm soll vom Benutzer 2 Geldbeträge und 1 Zahl einlesen, und damit die Funktionen der Addition, Subtraktion, Skalierung und Vergleich demonstrieren.

Bonuspunkt (+1)

Ermögliche die Konvertierung von Geldbeträgen mit einer Methode und einer internen Konversionstabelle (z.B. RON -> EUR, RON -> HUF)

Labor 2

Diskussion: Kalenderw. 12, **Abgabe:** Kalenderw. 13, **Punkte:** 4/55

Bonuspunkt (+1)

Ermögliche die übliche infix-Schreibweise für die Operationen +, -, *, /, zusätzlich zu den Methoden.

Problem 2

Entwickeln Sie eine Klasse zur Darstellung und Verarbeitung von **Zeitdauern**. Folgende Dateien sind im Projekt zu erzeugen:

- `PREFIX_duration.h` enthält die Definition der Klasse
- `PREFIX_duration.cpp` enthält die Implementierung
- `PREFIX_duration_test.cpp` enthält `main()` mit **Testroutinen für jede Methode**

Ein Objekt für eine Zeitdauer sollte sowohl eine Fliesskommazahl für die Größe als auch einen String für die Einheit beinhalten. Auf diese Attribute sollte außen kein Zugriff erlaubt sein. Außerdem sollten alle Methoden sicherstellen, dass nur kompatible Zeitangaben miteinander verarbeitet werden (z.B. 3,6min + 7,4min = 11min oder 41,2h+56,5h=97,7h)

1. Schreiben Sie die Klasse ***Duration*** mit den erforderlichen Datenkomponenten, einem spezifizierten Konstruktor sowie den Zugriffsmethoden `get_value`, `get_unit`.
2. Erstellen Sie die Methoden `add`, `substract` so, dass die Datenkomponenten des Objektes unverändert bleiben, der zweite Operand als Parameter übergeben wird und der Rückgabewert das Ergebnis der Operation enthält.
3. Erstellen Sie eine `scale` Methode, welche eine Dauer mit einer Zahl multipliziert.
4. Erstellen Sie eine `divide` Methode, welche eine Dauer durch eine Zahl dividiert.
5. Erstellen Sie eine `text` Methode, welche die Zeitdauer mit Einheit als Zeichenkette darstellt.
6. Erstellen Sie eine `compare` Methode, welche zwei Zeitdauern vergleicht, wobei dur1.compare(dur2) folgende Rückgabewerte hat:

   -1 für dur1<dur2, 0 für dur1=dur2, +1 für dur1>dur2
7. Das Programm soll vom Benutzer 2 Zeitdauern und 1 Zahl einlesen, und damit die Funktionen der Addition, Subtraktion, Skalierung und Vergleich demonstrieren.

Bonuspunkt (+1)

Ermögliche die Konvertierung von Zeitdauern mit einer Methode und einer internen Konversionstabelle (z.B. min -> h, min -> sec)

Bonuspunkt (+1)

Ermögliche die übliche infix-Schreibweise für die Operationen +, -, *, /, zusätzlich zu den Methoden.