Kontenverwaltung

Schreiben Sie eine Klasse **`Barwert`**, die Geldwerte in unterschiedlichen Währungen repräsentieren kann. Es ist zumindest jeweils eine Instanzvariable für die Währung und für den Betrag vorzusehen. Der Betrag darf nicht negativ werden. Schreiben Sie weiters eine Klasse **`Konto`** zur Repräsentation eines (einfachen) Bankkontos. Ein Konto kann in einer beliebigen Währung (die nach dem Erstellen des Kontos nicht mehr verändert werden kann) geführt werden und hat einen aktuellen Kontostand, sowie einen Dispo-Rahmen. Für jedes Konto werden die Beträge aller Buchungen gespeichert, sodass diese später in Form eines „Kontoauszugs" ausgegeben werden können. Folgende Methoden und Operatoren müssen in der Klasse **`Konto`** implementiert werden:

- Operator +=: Die Verknüpfung eines Kontos und eines Barwerts mit += soll den Wert auf dem Konto gutschreiben. Dabei ist gegebenenfalls eine Währungskonversion vorzunehmen.
- Operator -=: Analog zu Operator +=, allerdings ist der Wert vom Konto abzubuchen. Kann die Abbuchung nicht durchgeführt werden, so soll eine Exception vom Typ **`runtime_error`** geworfen werden.
- Operator ==: Vergleicht die Werte zweier Konten (die Dispo-Rahmen sind für das Ergebnis irrelevant). Gegebenenfalls muss eine Währungskonversion durchgeführt **`werden`**.
- Methode **`bool umbuchen(double betrag, Konto& von)`**: Bucht den angegebenen Betrag (dieser ist in der Währung des Kontos zu verstehen, für welches die Methode aufgerufen wird) vom Konto **`von`** auf das Konto, für welches die Methode aufgerufen wird. Gegebenenfalls muss eine Währungskonversion durchgeführt werden. Ist der Betrag negativ, so soll die Buchung in der anderen Richtung durchgeführt werden. Würde die Transaktion dazu führen, dass eines der beteiligten Konten über den Dispo-Rahmen hinaus überzogen wird, so ist die Buchung nicht durchzuführen und der Wert **`false`** zurückzuliefern. Im Erfolgsfall soll **`true`** retourniert werden.
- Methode **`void kontoauszug(unsigned anz_Buchungen=10) const`**: Es sollen die Beträge der letzten **`anz_Buchungen`** Buchungen auf dem Konto, für welches diese Methode aufgerufen wurde, durch Komma getrennt ausgegeben werden. Ist **`anz_Buchungen`** gleich 0 oder größer als die Anzahl der gespeicherten Buchungen, so sind die Beträge aller Buchungen auszugeben.

Implementieren Sie die Klassen **`Barwert`** und **`Konto`** mit den notwendigen Konstruktoren, Methoden und Operatoren, so dass jedenfalls das Rahmenprogramm (barwert.h, barwert.cpp, konto.h, konto.cpp, kontotest.cpp verfügbar unter /home/Xchange/ue9) kompiliert und ausgeführt werden kann und die gewünschten Ergebnisse liefert. Achten Sie in Ihren Konstruktoren darauf, dass nur gültige Barwerte und Konten erstellt werden können (Betrag in **`Barwert`** darf nicht negativ sein, **`Konto`** darf nicht über den Dispo-Rahmen hinaus belastet sein und der Dispo-Rahmen darf nicht negativ sein). Werfen Sie gegebenenfalls eine Exception.

Für Ihr Programm dürfen Sie **nur** die im vorgegebenen Rahmenprogramm angeführten include-Dateien verwenden!

Instanzvariablen sind **`private`** zu definieren und die Verwendung globaler Variablen ist (abgesehen von im Rahmenprogramm eventuell bereits definierten) nicht erlaubt! Interpretationsspielraum in der Angabe können Sie zu Ihren Gunsten nutzen.

Datei barwert.h:

```
#ifndef BARWERT_H
#define BARWERT_H

#include<iostream>
#include<vector>
#include<string>
using namespace std;

//Waehrungssymbole
enum class Waehrung {USD, EUR, CHF, CAD};

class Barwert {
  //Wechselkurse relativ zu USD in der Reihenfolge USD, EUR, CHF, CAD
  static const vector<double> wechseltabelle;
  //Waehrungsbezeichnungen in der Reihenfolge USD, EUR, CHF, CAD
  static const vector<string> waehrung_bezeichnungen;
};
#endif
```

Datei barwert.cpp:

```
#include<iostream>
#include<vector>
#include<string>
#include<stdexcept>
#include"barwert.h"
using namespace std;

const vector<double> Barwert::wechseltabelle {1, 0.94 , 1.02, 1.31};
const vector<string> Barwert::waehrung_bezeichnungen {"USD", "EUR", "CHF", "CAD"};
```

Datei konto.h

```
#ifndef KONTO_H
#define KONTO_H

#include <iostream>
#include <vector>
#include <string>
#include "barwert.h"
using namespace std;

//Definition der Klasse Konto
#endif
```

Datei konto.cpp

```
#include <iostream>
#include <vector>
#include <string>
#include<stdexcept>
#include "barwert.h"
#include "konto.h"
using namespace std;
```

Datei kontotest.cpp

```
#include<iostream>
#include<vector>
#include<string>
#include<stdexcept>
#include"barwert.h"
#include"konto.h"
using namespace std;

int main() {
  Barwert wert {Waehrung::USD, 250};
  try {
    Barwert {Waehrung::USD, -10};
  }
  catch(runtime_error& e) {
    cout << e.what() << '\n';
  }
  try {
    Konto {Waehrung::EUR, 100, -10};
  }
  catch(runtime_error& e) {
    cout << e.what() << '\n';
  }
  try {
    Konto {Waehrung::EUR, -500, 10};
  }
  catch(runtime_error& e) {
    cout << e.what() << '\n';
  }
  cout << Konto {Waehrung::EUR,-500,1000} << '\n';
  Konto k {Waehrung::EUR, 250, 1000};
  cout << k << '\n';
  ((k += wert) += wert) -= wert;
  cout << k << '\n';
  Konto k1 {Waehrung::EUR, 250, 1000};
  Konto k2 {Waehrung::USD, 10, 1000};
  if (k1.umbuchen(250,k2))
    cout << k1 << ',' << k2 << '\n';
  if (k1.umbuchen(1000,k2))
    cout << k1 << ',' << k2 << '\n';
  if (k1.umbuchen(-250,k2))
    cout << k1 << ',' << k2 << '\n';
  if (k1.umbuchen(-1251,k2))
    cout << k1 << ',' << k2 << '\n';
  if (k1 == k2) cout << "k1 gleich k2\n";
  Konto k3 {Waehrung::USD, 0, 1000};
  k3 += Barwert {Waehrung::EUR, 350};
  if ((k1 += Barwert {Waehrung::EUR, 100}) == k3) cout << "k1 gleich k3\n";
  cout << k3 << '\n';
  Barwert bw; //Standard ist 10 EUR
  Konto k4;  //Standard ist 0 EUR mit Rahmen 500
  Konto k5 {Waehrung::USD}, k6 {bw};
  cout << bw << '\n' << k4 << '\n' << k5 << '\n' << k6 << '\n';
  k1.kontoauszug();
  cout << '\n';
  k1.kontoauszug(2);
  cout << '\n';
  k1.kontoauszug(10);
  cout << '\n';
  return 0;
}
```

Gewünschte Ausgabe:

```
Negative Betraege in Barwert nicht erlaubt
Negativer Dispo nicht erlaubt
Konto ueberzogen
[EUR -500/1000]
[EUR 250/1000]
[EUR 485/1000]
[EUR 500/1000],[USD -255.957/1000]
[EUR 250/1000],[USD 10/1000]
k1 gleich k3
[USD 372.34/1000]
10 EUR
[EUR 0/500]
[USD 0/500]
[EUR 10/500]
250, -250, 100
-250, 100
250, -250, 100
```