Technische Hochschule Georg Agricola

Allgemeine Angaben

Modulprüfung	Klausur	Datum: 23.09.2021
Modulname: **Objektorientierte Programmierung**	Modulnummer:	40050200
Prüfungsdauer: **60 min + 15 min**	Prüfer: Welp	

Name: _____ Vorname:_____

Matrikelnummer: _____

Bewertung

Aufgabe	1	2	3	4						Summe
Erreichbare Punkte	11	13	15	15						54
Erzielte Punkte										
Unterschrift Prüfer				Erzielte Punkte in %					Note	

© Prof. Dr. Welp, 2016

Aufgabe 1: Von C nach C++

Gegeben sei folgender Datentyp:

```
struct ComplexT{
   double r,i;
};
```

1. Schreiben Sie eine Funktion `phase` zur Berechnung der Phase einer komplexen Zahl. Die komplexe Zahl soll über eine Parameter `z` des Typs `ComplexT` übergeben werden. Die Funktion soll die Phase wahlweise in Grad oder Bogenmaß zurückliefern. Dies soll über einen zweiten Parameter `einheit` vom Typ `int` gesteuert werden (0: Bogenmaß, 1: Grad). Die Angabe der Einheit soll beim Funktionsaufruf optional sein. Wird sie weggelassen, soll mit Bogenmaß gerechnet werden.

   Lösen Sie dieses Problem, indem Sie

   1. `phase` überladen
   2. `phase` als Funktion mit Defaultparametern definieren.

   Geben Sie für beide Lösungswege die Funktionsdefinitionen an. (7 Punkte)

   Hinweis: Zur Berechnung der Phase kann die `atan2`-Funktion der C-Standardbibliothek verwendet werden.

2. Definieren Sie eine Funktion `umdrehen`, die von einer als **Referenz** übergebenen `ComplexT`-Variable das Vorzeichen des Realteils umdreht. Definieren Sie ferner eine `main`-Funktion, in der die Funktion `umdrehen` getestet wird. (4 Punkte)

© Prof. Dr. Welp, 2016

Aufgabe 2: Klassen und Objekte

Definieren Sie eine Klasse `OrtsVec` zum Speichern und Rechnen mit Ortsvektoren mit folgenden Eigenschaften *(Anm.: Lassen Sie in der Klasse genügend Platz um nach und nach Ergänzungen vornehmen zu können)*:

a) Der direkte Zugriff auf die Attribute eines Vektors soll nicht möglich sein. Der Zugriff soll über entsprechende setter- und getter-Methoden erfolgen. Implementieren Sie die Methoden als inline-Funktionen.

b) Die Klassse soll über geeignete Kontruktoren zur Initialisierung von `OrtsVec` -Objekten verfügen, die eine Objektinstanziierung entsprechend Hinweis 1 ermöglichen. Bei der Instanziierung eines `OrtsVec` -Objektes ohne Parameter, sollen die Komponenten des Vektors mit 0 initialisiert werden. Implementieren Sie die Methoden als inline-Funktionen.

c) Über eine Methode soll die Länge eines `OrtsVec` -Objektes abgefragt werden können. Implementieren Sie diese Methode ausserhalb der Klassendefinition. Fügen Sie in das Programmfragment von Hinweis 1 eine Anweisung ein, mit der die Länge des Vektors `v1` ausgegeben wird.

d) Die Klasse soll über ein Klassenattribut verfügen, in welchem die Anzahl der `OrtsVec`-Objekte gespeichert wird (Objektzähler). Die Anzahl der `OrtsVec` -Objekte soll über eine Klassenmethode abfragbar sein.

e) Ferner soll der **^**-Operator für die Klasse `OrtsVec` überladen werden. Der **^**-Operator soll das Skalarprodukt zweier Ortsvektoren ermitteln. **Deklarieren** Sie den Operator. *(siehe Hinweis)*

(13 Punkte)

Hinweise

1. Die Klasse Vektor soll z.B. folgendermaßen verwendet werden können

```
int main()
{
   OrtsVec v1(1,2,3),v2;
   v2.setX(4);
   v2.setY(5);
   v2.setZ(6);

   cout << "Skalarprodukt von v1 und v2 ist: " << v1^v2 << endl;
   cout << "Es gibt insgesamt " << Vektor::getAnzahl() <<" Vektoren";

}
```

Ausgabe:

```
Skalarprodukt von v1 und v2 ist: 32
Es gibt insgesamt 2 Vektoren.
```

2. Die Länge eines Vektors berechnet aus $\sqrt{x^2 + y^2 + z^2}$

© Prof. Dr. Welp, 2016

Aufgabe 3: Objektbeziehungen und Templates

In der Bibliotheks-Software einer Uni existieren die Klassen `Student` und `Buch`. Zwischen Objekten dieser beiden Klassen besteht folgende Assoziation („Student reserviert Buch“):


```
class Student{
   string name;
pubic:
   Student(string n):name(n){};
   string getName(){return name;};
};
```

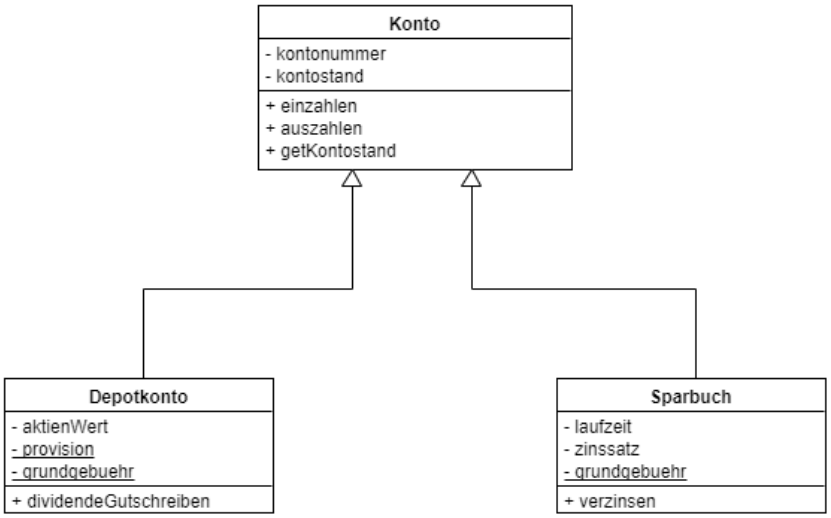
```
class Buch{
   string titel;
pubic:
   Buch(string t):titel(t){};
   string getTitel(){return titel;};
};
```

1. Ergänzen Sie die Definition für die Klasse `Student`, sodass die Assoziation zur Klasse `Buch` implementiert wird. Es sollen Methoden existieren, mit dem ein Buch reserviert werden kann, mit dem eine Buchreservierung aufgehoben werden kann und mit der Reservierungen abgefragt werden können. Beschreiben Sie kurz die Aufgabe, die Parameter und den Rückgabewert der Methoden! *Die Methoden brauchen nicht implementiert werden.* (8 Punkte)
2. Schreiben Sie ein Hauptprogramm, in dem eine Container-Variable zur Speicherung von Studentenobjekten definiert wird. **Verwenden Sie hierzu eine Container-Klasse der Standard Template Library (STL).** (1 Punkte)
   Fügen Sie der Variable einige (mindestens 3) Studentenobjekte hinzu. (2 Punkte)
   Ergänzen Sie Ihr Programm um Anweisungen zur Ausgabe des Namens aller gespeicherten Studenten. (4 Punkte)

© Prof. Dr. Welp, 2016

Aufgabe 4: Vererbung und Polymorphismus

1. Gegeben sei folgendes Klassendiagramm:



Die Klasse `Konto` ist wie folgt definiert:

```
class Konto
{
    unsigned int kontonummer;
    float kontostand;
public:
    Konto(unsigned int knr, float ks=0.0.);
    float getKontostand();
    void einzahlen(float b);
    void auszahlen(float b);
};
```

a) Geben Sie Klassendefinition für die Klasse `Depotkonto` an. (4 Punkte)

b) Über welche Attribute verfügen **Objekte** der Klasse `Depotkonto`? (3 Punkte)

c) Über welche Methoden verfügen **Objekte** der Klasse `Depotkonto`? (2 Punkte)

2. Welchen Aussagen zum Observer-Entwurfsmuster stimmen Sie zu? (2 Punkte)

- ☐ Wenn sich die Schnittstelle eines Subjektes ändert, hat das auch Auswirkungen auf den Beobachter.
- ☐ Bevor ein Beobachter über Zustandsänderungen eines Subjekts informiert wird, muss es sich bei dem Subjekt registrieren.
- ☐ Subjekte aktualisieren ihren Zustand sobald sich Beobachter bei den Subjekten registrieren.
- ☐ Ein Beobachter ruft sporadisch die Methode `notify()` des Subjekts auf, um über Zustandsänderungen des Subjekts informiert zu sein.
- ☐ Der Beobachter muss eine definierte Schnittstelle implementieren, die in einer abstrakten Basisklasse definiert ist, von der der Beobacher abgeleitet wird.

© Prof. Dr. Welp, 2016