Übung 4

Container, Überladen von Operatoren, klassenspezifische Daten

Aufgabe 4.1 (klassenspezifische Daten, Operatoren)

(a) Welche Änderungen müssen Sie an Ihrer Implementierung der Klasse **Pkw** vornehmen, damit die Fahrgestellnummer fortlaufend ab der Nummer 1000 vergeben wird?

(b) Erweitern Sie Ihre Implementierung um einen überladenen Operator für die Klasse Pkw, so dass folgendes Programmstück ohne Fehler abläuft:

```
if (p1 < p2)     cout << "p1 ist kleiner als p2" << endl;
else             cout << "p2 ist kleiner gleich p1" << endl;
```

Dabei soll p1 < p2 sein, falls die Fahrgestellnummer von p1 kleiner als die von p2 ist.

Aufgabe 4.2 (Überladen von Operatoren)

Implementieren Sie eine Klasse **`complex`**, die die komplexen Zahlen realisiert und die dafür gängigen Operatoren **`+,-`** und **`*.`**

Aufgabe 4.3 (Container)

Schreiben Sie mit Hilfe des Standard-Containers `vector` eine Programm, das das Vorkommen einzelner Wörter in einem Eingabetext (von der Standardeingabe) zählt

Also, die Eingabe
```
aa  bb bb aa aa bb aa aa
```
erzeugt folgende Ausgabe
```
aa: 5
bb: 3
```

Verwende Sie die folgende Datenstruktur

```
struct paar{
     string word;
     int count;
};
```

Aufgabe 4.4 **(klassenspezifische Daten, friend, Operatoren**)

(a) Welche Änderungen müssen Sie an Ihrer Implementierung der Klasse **Student** vornehmen, damit die **Matrikelnummer** fortlaufend ab der Nummer 1000000 vergeben wird?

(b) Hat Ihre Klasse Student einen Defaultkonstruktor? Wenn nein, dann ergänzen Sie die Klasse um einen Defaultkonstruktor z.B. indem Sie einen Default-Wert für den Namen vorsehen.

(c) Was ändert sich an Ihrer Implementierung, wenn Sie die Klasse **NotenListe** als `friend` der Klasse **Note** deklarieren?

(d) Was müssen Sie ändern, wenn Sie die NotenListe nicht als verkettete Liste, sondern als lineares Array implementieren?