# Überladung von Konstruktoren und Operatoren

## AUFGABE 3.1 (Ausgabe)

Nehmen Sie eine Ihrer alten Übungen und fügen Sie den erstellten Klassen einen Anzeigeoperator hinzu (z.B. die Klassen Kreis, Punkt3D, Dreieck, Artikel, Warenkorb, ...).

## AUFGABE 3.2 (Komplexe Zahlen) - **Abgabe**

Der Zweck dieser Aufgabe besteht darin, die KomplexeZahl-Klasse zu implementieren und die Operatoren zu definieren, die zum Schreiben einfacher arithmetischer Operationen mit komplexen und reellen Zahlen erforderlich sind.

(a) Definieren Sie die Klasse KomplexeZahl unter Verwendung der kartesischen Darstellung komplexer Zahlen (d. h. mit zwei double-Attributen, die jeweils den Real- und Imaginärteil der komplexen Zahl darstellen).

(b) Fügen Sie dieser Klasse die erforderlichen Konstruktoren und Destruktoren hinzu, damit die folgende Hauptklasse kompiliert werden kann:

```
int main()
{
      KomplexeZahl default_zahl;
      KomplexeZahl null_zahl(0.0, 0.0);
      KomplexeZahl eins(1.0, 0.0);
      KomplexeZahl i(0.0, 1.0);
      KomplexeZahl j;
      return 0;
}
```

(c) Definieren Sie die erforderlichen Operatoren (Sie benötigen zwei, einen internen und einen externen), damit diese Fortsetzung der Hauptfunktion korrekt kompiliert und ausgeführt werden kann:

```
cout << null_zahl << " ==? " << default_zahl;
if (null_zahl == default_zahl) cout << " ja" << endl;
else cout << " nein" << endl;
cout << null_zahl << " ==? " << i;
if (null_zahl == i) cout << " ja" << endl;
else cout << " nein" << endl;
```

(d) Fahren Sie mit einfachen arithmetischen Operatoren fort (auch hier werden externe Operatoren benötigt):

```
j = eins + i;
cout << eins << " + " << i << " = " << j << endl;
KomplexeZahl drei(eins);
drei += eins;
drei += 1.0;
cout << eins << " + " << eins << " + 1.0 = " << drei <<
endl;
KomplexeZahl zwei(drei);
zwei -= eins;
cout << drei << " - " << eins << " = " << zwei << endl;
drei = 1.0 + zwei;
cout << "1.0 + " << zwei << " = " << drei << endl;
```

(e) Fahren Sie dann mit der Multiplikation und Division fort:

```
KomplexeZahl z(i* i);
cout << i << " * " << i << " = " << z << endl;
cout << z << " / " << i << " = " << z / i << " = ";
cout << (z /= i) << endl;
```

(f) Und zum Schluss noch die wenigen Rechenoperatoren, die noch fehlen:

```
KomplexeZahl k(2.0,-3.0);
z = k;
z *= 2.0;
z *= i;
cout << k << " * 2.0 * " << i << " = " << z << endl;
z = 2.0 * k * i / 1.0;
cout << " 2.0 * " << k << " * " << i << " / 1 = " << z <<
endl;
```

Hinweis:

$$z_1 = (x_1, y_1) \text{ und } z_2 = (x_2, y_2)$$

$$z_1 \cdot z_2 = (x_1 \cdot x_2 - y_1 \cdot y_2, x_1 \cdot y_2 + y_1 \cdot x_2)$$

$$\frac{z_1}{z_2} = \left(\frac{x_1 \cdot x_2 + y_1 \cdot y_2}{x_2 \cdot x_2 + y_2 \cdot y_2}, \frac{y_1 \cdot x_2 - x_1 \cdot y_2}{x_2 \cdot x_2 + y_2 \cdot y_2}\right)$$

# AUFGABE 3.4 (Chemie) - **Abgabe**

Ein Chemiker benötigt Ihre Hilfe, um die Flakons (Glasfläschchen) der von ihm verwendeten Chemikalien zu modellieren. Laden Sie den auf StudIP verfügbaren Quellcode herunter und vervollständigen Sie ihn gemäß den nachstehenden Anweisungen. Der gelieferte Code beginnt mit der Deklaration der Klasse `Flakon`, definiert eine Hilfsfunktion `anzeige_mix` und deklariert schließlich in `main()` mehrere „Flakons" und zeigt deren Mischungen an.

(a) Der Hauptteil der Klasse Flakon fehlt und Sie müssen ihn bereitstellen.
Ein „Flakon" zeichnet sich aus durch:
- den Namen des darin enthaltenen Produkts, z.B. „Perchlorsäure";
- sein Volumen (in ml), vom Typ `double`;
- und seinen pH-Wert (vom Typ `double`).

Die Klasse `Flakon` enthält außerdem Folgendes:

(b) einen Konstruktor, der die Attribute mithilfe von als Parameter übergebenen Werten in derselben Reihenfolge initialisiert, wie es in der angegebenen `main()`-Anweisung gezeigt wird; Es soll keinen Standardkonstruktor geben.

(c) eine Methode `etikett`, mit dem Prototyp

```
ostream& etikett(ostream& ausgabe) const;
```

die das Etikett anzeigt, das auf dem „Flakon" angebracht werden soll. Dieses Etikett gibt den Namen, das Volumen und den pH-Wert des Inhalts des Flakons gemäß folgenden Anzeigeformats an:

```
Perchlorsäure: 250 ml, pH 2
```

(d) die Überladung des Anzeigeoperators << für diese Klasse; Für diese Überladung sollte die vorherige Methode `etikett` verwendet werden.

(e) Schließlich sollte es auch möglich sein, zwei „Flakons" mit dem Operator + zu mischen: gegeben sind zwei Flakons mit jeweiligen Namen, Volumina und pH-Werten: Name1, Volumen1, pH1 und Name2, Volumen2, pH2; Die Mischung dieser beiden Flakons mit dem Operator + ergibt einen neuen „Flakon", für den:

1. der Name „name1 + name2", ergibt z.B. „Wasser + `Perchlorsäure";`
2. Das Volumen ist die Summe von `volumen1` und `volumen2;`
3. Der pH-Wert ist (chemisch approximiert):

$$pH = -log_{10}\left(\frac{volumen1 \times 10^{-ph1} + volumen2 \times 10^{-ph2}}{volumen1 + volumen2}\right)$$

Die Funktion $log_{10}$ wird als `log10` in C++ geschrieben und um $10^{-x}$ zu berechnen nutzen Sie `pow(10.0, -x)`.

(f) Stellen Sie für die Klasse Flakon einen internen Operator `operator+=` zur Verfügung. Dieser ermöglicht die Mischung des „Flakons". Auch wenn dies physikalisch nicht korrekt ist (deshalb verwenden wir Anführungszeichen), kann ein solcher „Flakon" sein Volumen wie unter (e) beschrieben erhöhen; D.h. nach a += b; verändert sich das Volumen von `a` (dies ist auch der Fall bei seinem Namen und seinem pH-Wert).

**Ausgabebeispiel:**

```
Mische ich
"Wasser : 600 ml, pH 7"
mit
"Salzsäure : 500 ml, pH 2"
Erhalte ich :
"Wasser + Salzsäure: 1100 ml, pH 2.34242"
Mische ich
"Salzsäure : 500 ml, pH 2"
mit
"Perchlorsäure: 800 ml, pH 1.5"
Erhalte ich :
" Salzsäure + Perchlorsäure: 1300 ml, pH 1.63253"
```