

## Klassen §9

Motivierung am Bsp. Graph

*graph* Klassisch  $\rightarrow$  *graph\_3* objektorientiert

- Eigenschaften (engl. properties) der Klasse
  - Daten  $\rightarrow$  Member der Klasse
  - Funktionen  $\rightarrow$  Methods der Klasse
- Variable (`double p(0.2);`)  
 $\rightarrow$  Instanz/Objekt der Klasse (`graph g1("g_2.txt");`)
- Datenkapselung (engl. encapsulation) via `private` (`protected`, `public`)
- Vererbung (engl. inheritance): Basisklasse, abgeleitete Klassen, Klassenhierarchie  
IS-A und HAS-A Eigenschaften abgeleiteter Klasse.
- virtuelle Methoden: abstrakte und konkrete Klassen
- Polymorphismus

Eigene Klasse implementieren:

- Grobentwurf der Klasse: Member (Daten) , Methoden (Funktionalität)
- Codeblocks: File  $\rightarrow$  New  $\rightarrow$  Class. [kurze Demo]  
 $\rightarrow$  *komplex.h*, *komplex.cpp*

### Klasse Komplex

Komplexe Zahlen gibt es bereits in `<complex>`, die eigene Klasse dient zur Demonstration der Implementierungsdetails dahinter.

Was soll die eigene Klasse `Komplex` für  $x + yi$  enthalten?

- Realanteil  $x$ , Imaginäranteil  $y$  Member
- Deklaration ohne Initialisierung: `Komplex b` parameterloser Konstruktor
- Deklaration mit Initialisierung: `Komplex a(2.3,4.2)` Parameterkonstruktor
- Deklaration mit Initialisierung aus Realteil: `Komplex a(2.3)` Parameterkonstruktor
- Deklaration mit Objektinit.: `Komplex c(a)` Copykonstruktor
- Zuweisung: `b = a` Zuweisungsoperator `operator=` [Method]
- Addition: `a+b` Operator `operator+` [Method/Function]
- Ausgabe: `cout << c` Ausgabeoperator `operator<<` [Function]
- Getter-/Settermethoden
- Betrag: `abs(c)` [Method]

Für Lecturer:

Verzeichnis *v\_6a/* in *v\_6a\_demo/* kopieren und *main.cpp*:11-27 kommentieren, desgleichen *komplex.cpp*:7-76.

Danach schrittweise die Funktionalität zuschalten.

Weiter wie im Skript §9.

Compiler option `-Weffc++`<sup>1</sup>.

- Member: Real-, Imaginäranteil
- Konstruktoren (mehrere!):
  - Member Initialization List
  - ohne Parameter (Standardinitialisierung!?)
  - mit Parameter(n); optionale Argumente?
  - Copy-Constructor
  - [Move-Constructor]
  - Rule-of-Five<sup>2</sup>
- Destruktor (genau einer)
- Zuweisungsoperator
  - Copy-
  - [Move-]
- Methoden:
  - Setter
  - Getter (const Method)
  - `operator+` für `a+b` oder `a+2` (const Method)
  - `operator+=`
  - `abs` (const Method)
- Funktionen:
  - Ausgabe: `operator<<`
  - Eingabe: `operator>>`
  - `operator+` für `2+a`

Aktuelles C++17 in `v_6a_cpp17/`

---

Fun with ASCII<sup>3</sup>: `char_sum.cpp`

- How to achieve 103% performance?
- { HARDWORK, KNOWLEDGE, ATTITUDE, BULLSHIT };
- `magic=96` → `magic=0` : BILLGATES (the third!)

## Literatur

[Haase] Gundolf Haase: Einführung in die Programmierung mit C++ (2023), *www*<sup>4</sup>.

[Stroustrup10] Bjarne Stroustrup: Einführung in die Programmierung mit C++. Pearson Studium, München (2010).

---

<sup>1</sup><https://www.intel.com/content/www/us/en/develop/documentation/cpp-compiler-developer-guide-and-reference/top/compiler-reference/compiler-options/compiler-option-details/compiler-diagnostic-options/weffc-qeffc.html>

<sup>2</sup>[https://en.cppreference.com/w/cpp/language/rule\\_of\\_three](https://en.cppreference.com/w/cpp/language/rule_of_three)

<sup>3</sup><http://www.torsten-horn.de/techdocs/ascii.htm>

<sup>4</sup>[http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Script/html/script\\_programmieren.pdf](http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Script/html/script_programmieren.pdf)