

## Klassen §9

### Klasse Komplex cnt.

Rekapitulation zu Beispiel *v\_6a*:

- Member: Real-, Imaginäranteil
- Konstruktoren (mehrere!):
- Destruktor (genau einer)
- Zuweisungsoperator
- Methoden (Funktionen der Klasse):
  - Setter
  - Getter (const Method)
  - `operator+` für `a+b` oder `a+2` (const Method)
  - `operator+=`
  - `abs` (const Method) ← 8. April
- Member (Daten der Klasse)
  - **private** → Datenkapselung (encapsulation)
- Funktionen:
  - Ausgabe: `operator<<`
  - Eingabe: `operator>>`
  - `operator+` für `2+a`
  - allgemein: binärer Operator (wie `operator+`) als Funktion, aber `operator+=` als Methode implementieren.
    - \* Funktion nutzt Getter/Setter für Zugriff auf Member, oder als
    - \* **friend**-Funktion in Klasse deklarieren.

Demo `vector<Komplex>` mit parameterlosem Konstruktor und zu Vektorausgabe am Beispiel *v\_6a\_vektor*:

- `vector<Komplex> vv;` ← `Komplex::Komplex()` wird benötigt.
- Demo der (Nicht-)Initialisierung im parameterlosem Konstruktor.
- Initialisierung des Vektors mit initializer list mit Elementen `Komplex(1.1,2.3)` oder `{1.1,2.3}` .
- Ausgabeoperator (function) für `vector` (allg. mit Templates).

## Aktuelles C++17 in *v\_6a\_cpp17/*

- Rule of Five via `= default`
- Constructor forwarding
- Header Guarding: `#pragma once`
- `vector<Komplex>`; Vektorausgabe; Name der Variablen (Macro)
- `sort(#include <algorithm>)` benötigt `bool operator<(Komplex a, Komplex b) .`
- `sort` über  $\lambda$ -Funktion als Sortierkriterium: `[ ]( ){ }`
  - `[ ]` **Caption**, d.h. Transfer von gültigen Variablen/Instanzen in die Funktion.
  - `( )`: übliche **Parameterliste**
  - `{ }`: Funktionskörper
  - `[ ](Komplex lhs, Komplex rhs) -> bool { return lhs.abs()<rhs.abs(); }`

### zu **Aufg. 19**: $\lambda$ -Funktion *Bisect/Bisect\_3\_lambda.cpp*

- `double f(const double x) { return sin(x) - 0.5*x; }` wird in `double x0 = Bisect3(f, a, b, EPS);` als auszuwertende Funktion übergeben.
- dasselbe als  $\lambda$ -Funktion:  
`double x0 = Bisect3([ ](const double x) { return sin(x) - 0.5*x; }, a, b, EPS);`

Jetzt wollen wir ein Polynom  $p(x) = \sum_{k=0}^n a_k x^k$  an `Bisect` übergeben:

- Gegeben: Auswertungsfunktion für  $p(x)$   
`double eval(vector<double> const &a, double x)`
- Wie übergeben wir den zusätzlichen Koeffizientenvektor  $a$  an `Bisect`?  
==> via **Caption** in der  $\lambda$ -Funktion.
- `double x0 = Bisect3([a](const double x) { return eval(a,x); }, a, b, EPS);`

## Klasse **Student** *v\_7a/*

Entwurf:

- Member: Familienname, Matrikelnr. (const?), Studienkennzahl (mehrere)?
- Methoden:
  - Konstruktor(en); Nutzung `default/delete`
  - Ausgabeoperator
  - Getter/Setter (Namensänderung?)
  - Hinzufügen/Entfernen von Studien
- `const` Member  $\implies$  `operator=` nicht mehr möglich.
- Getter sind `const` Methoden.

- `int _skz`  $\rightarrow$  `vector<int> _skz`  
 $\Rightarrow$  `const vector<int>& Get_SKZ() const {return _skz;}`  
 $\Rightarrow$  zusätzlicher Konstruktor
- `Add_SKZ`: Prüfung auf vorhandenes Studium + `push_back`
- `Del_SKZ`: Prüfung auf vorhandenes Studium + `erase`
- Nutzung von `set` statt `vector` erspart das Suchen in `_skz`.  
 $\Rightarrow$  `v_7a_set/`: Hauptprogramm bleibt **unverändert**.

### Nutzung der STL mit Container der Klasse Student *v\_7b/*

---

Fun with ASCII<sup>1</sup>: *char\_sum.cpp*

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

## Literatur

[Haase22] Gundolf Haase: Einführung in die Programmierung mit C++ (2022), *www*<sup>2</sup>.

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

---

<sup>1</sup><http://www.torsten-horn.de/techdocs/ascii.htm>

<sup>2</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)