Draft der C++-Vorlesung vom 8.April 2022

Klassen §9

Motivierung am Bsp. Graph
graph Klassisch $\longrightarrow$ *graph_3* objektorientiert

- Eigenschaften (engl. properties) der Klasse
  - Daten $\longrightarrow$ Member der Klasse
  - Funktionen $\longrightarrow$ Methods der Klasse
- Variable (`double p(0.2);`)
  $\longrightarrow$ 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 → New → Class. [kurze Demo]
  $\longrightarrow$ *komplex.h. komplex.cpp*

Klasse Komplex

Komplexe Zahlen gibt es bereits in `<complex>`, die eigene Klasse dient zur Demonstration der Implementieriungsdetails 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++`[1].

- Member: Real-, Imaginäranteil
- Konstruktoren (mehrere!):
  - Member Initialization List
  - ohne Parameter (Standardinitialisierung!?)
  - mit Parameter(n); optionale Argumente?
  - Copy-Constructor
  - [Move-Constructor]
  - Rule-of-Five[2]
- 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[3]: *char_sum.cpp*

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

Literatur

[Haase22] Gundolf Haase: Einführung in die Programmierung mit C++ (2022), *www*[4].

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

[1] `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`
[2] `https://en.cppreference.com/w/cpp/language/rule_of_three`
[3] `http://www.torsten-horn.de/techdocs/ascii.htm`
[4] `http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Script/html/script_programmieren.pdf`