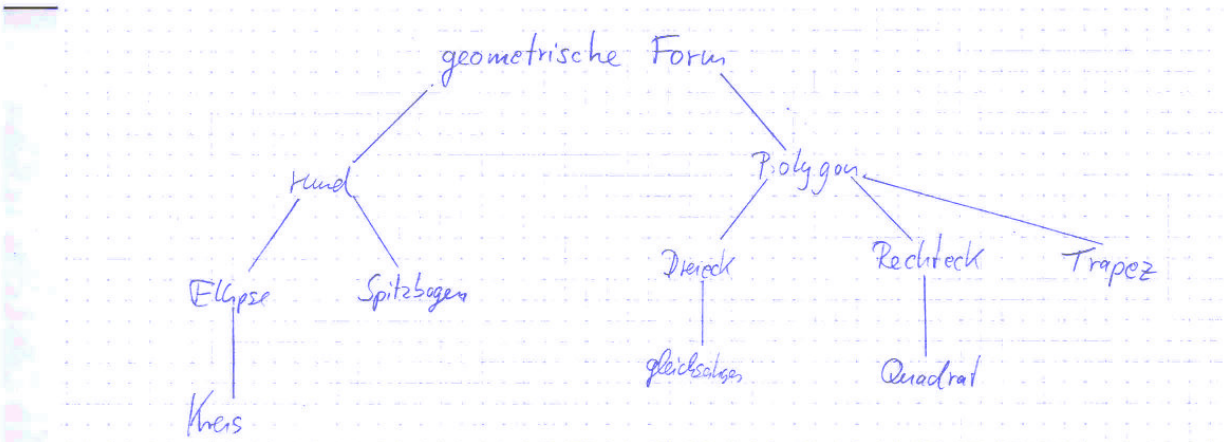
6. Übung des Programmierpraktikums

Abgabetermin: 27. Juni 2022, 23:59 Uhr

Die Übungen sind grundsätzlich selbst zu lösen.
Zweiergruppen sind erlaubt, aber nur unter Angabe des Partners im gut sichtbaren Kommentar. Speichern Sie die Funktionsdeklarationen jeweils in einem separaten Headerfile und die Funktionsdefinitionen in einem separaten Sourcefile, d.h., jedes Projekt enthält mindestens zwei **.cpp* und ein **.h* File(s). Alle eigenen Funktionen und Klassen sind so im Headerfile zu dokumentierten, daß die Funktionalitäten für Dritte ausreichend beschrieben werden und daß `doxygen` damit arbeiten kann.

---

25. Entwerfen, implementieren und testen Sie eine *Klassenhierarchie*[1] analog zum, nicht zwingenden, Vorschlag[2]:



Folgende Funktionalität soll in Ihrer Hierarchie vorhanden sein: (6 Pkt.)

- Die übergeordnete Basisklasse soll eine **abstrakte Klasse** sein.
- Diese Basisklasse soll eine Farbinformation speichern. Hier können Sie auch den Aufzählungstyp (`enum`) aus §5.5. des Skriptums benutzen.
- Neben den nötigen Konstruktoren sollen die Klassen Methoden zur Abfrage des Namens der Klasse, des Umfanges und der Fläche vorhanden sein.
- Die Kategorisierung *rund* oder *gerade* für die Formumrandung soll abfragbar sein.
- Die Basisklasse soll eine Farbinformation speichern, z.B.:

```
//    Radius, Farbe
Kreis kk(5.5, rot);
//         a,   b,   c, beta, Farbe
Trapez tt(5.5, 3.0, 2.5, 60, blau);
cout << kk << endl;
```

- Details zu den Formen unter Geometrierechner[3].

[1] `http://www.cplusplus.com/doc/tutorial/`
[2] `https://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS22/bsp_25_hierarchy.jpg`
[3] `https://rechneronline.de/pi/formen.php`

- Implementieren Sie auch einen Ausgabeoperator für eine Referenz auf die Basisklasse welcher alle Eigenschaften des konkreten Objektes ausgibt.

Hinweis: Falls unter Windows die Konstante `M_PI` vom Compiler nicht erkannt wird, dann vor der Zeile `#include <cmath>` die Zeile `#undef __STRICT_ANSI__` einfügen.

26. *Polymorphismus*[4]: Erzeugen Sie sich einen Container (Vektor/Liste von Basisklassenpointern) mit mind. 16 Formen, wobei jeder Ihrer mind. 8 Formenklassen aus Beispiel 25 zumindest doppelt vorhanden sein soll. Verwenden Sie dabei 3 Farben. Die Basisklassenpointer können klassische C-Pointer sein, besser aber `shared_ptr<Basisklasse>` bzw. `unique_ptr<Basisklasse>`. (6 Pkt.)

    - Lösen Sie mittels der Algorithmen aus der STL und geben Sie anschließend immer das Ergebnis aus:
        (a) Aufsteigendes Sortieren der Formen nach Umfang.
        (b) Absteigendes Sortieren der Formen nach Flächeninhalt.
        (c) Finden Sie die Form mit dem größten Umfang.
        (d) Finden Sie die flächenmäßig kleinste *blau*e Form.
        (e) Entfernen Sie die größte runde Form.
        (f) Kopieren Sie alle *rot*en *gerad*en Formen in einen neuen Container, siehe §11.3.4 im Skript bzw. `copy_if`[5] oder www[6].
        (g) Bestimmen Sie den Gesamtumfang aller Formen.
        Hinweis: For-Loop (langweilig) oder `accumulate` benutzen.
        (h) Entfernen Sie alle *geraden* Formen deren Fläche größer ist als die mittlere Fläche der *runden* Formen.
    - (*) Entfernen Sie alle mehrfach vorhanden, identischen Formen aus dem originalen Container.
      Tip: Zum Ordnen der Datentypen kann man `before`[7] in Verbindung mit `typeid`[8] benutzen.
    - (*) Benutzen Sie `operator<<` um alle geometrischen Formen in ein Textfile zu schreiben.

Hinweise:
Die Schlüsselwörter `continue, goto` dürfen nicht benutzt werden, `break` nur in Verbindung mit der `switch`-Anweisung. Desgleichen darf ein `return` nur als letzte Anweisung einer Funktion vorkommen.

Die Abgabe der Lösungen (`*.cpp`-Files, `*.h`, (`Makefile*`,) ...,) erfolgt an der KFU über Moodle, siehe dazu die Hinweise auf der LV-Homepage[9].

Keine Leerzeichen, Sonderzeichen oder Umlaute in File- und Verzeichnisnamen benutzen (Portabilität).

[4] `http://www.cplusplus.com/doc/tutorial/polymorphism/`
[5] `http://www.cplusplus.com/reference/algorithm/copy_if`
[6] `https://en.cppreference.com/w/cpp/algorithm/copy`
[7] `https://en.cppreference.com/w/cpp/types/type_info/before`
[8] `https://en.cppreference.com/w/cpp/language/typeid`
[9] `https://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS22/index.html`