ÜBUNGEN zu Programmiersprache 1 (Python, C++) WS 2021/2022

Übungsblatt Nr. 12: in der Übungsstunde

1) Wir starten mit der letzten `Bruch` Aufgabe und erhöhen die Sicherheit des Typs:
   a) Machen Sie `z, n private`. Sie werden einige Fehlermeldungen bekommen :-)
   b) Machen Sie ab den Konstruktoren alles `public`. Wie viele Fehlermeldungen gibt es jetzt weniger?
   c) Nehmen Sie alle `Bruch`-Funktionen in die Typdefinition auf. Wie viele Fehler jetzt?
   d) Schreiben Sie vor alle Funktionen `friend`. Und jetzt?

2) Nachdem das Programm wieder läuft, gehen wir `Bruch.hpp` durch:
   a) Der Programm-Abbruch in `normalize()` ist schlecht. Man wirft besser eine Exception, die vom Anwender bei Bedarf aufgefangen werden kann. Wir inkludieren `stdexcept` und werfen einen `runtime_error`.
   b) Diese Methode wird eigentlich nur von `Bruch` selbst gebraucht, kann also in den privaten Teil wandern.
   c) Das negative Vorzeichen kann man durchaus als Methode schreiben. Ich mache das bei unären Operatoren eigentlich immer.
   d) Umwandlungsfunktionen sollten eigentlich auch immer Operatoren sein. Machen Sie das mit `to_double()`.
   e) Zu diesem Zweck gibt es in C++ eigentlich Umwandlungs-Operatoren. Schreiben Sie in das C++-Programm `static_cast<double>(sum);` und testen es. Dann bauen Sie den Umwandlungsoperator im Header ein.
   f) Der Umwandlungsoperator im Header ist noch nicht gut. Um das zu sehen, berechnen Sie im C++-Programm `sum + 0`: Der Compiler beschwert sich über 2 gleichwertige Varianten: Er kann die 0 über den Bruchkonstruktor in einen Bruch umwandeln und die Bruchaddition verwenden. Oder kann den Bruch mit der neuen Methode in einen `double` umwandeln und die Zahlen-Addition nehmen. Umwandlungen in beide Richtungen haben oft diese Konsequenz! Machen Sie den Umwandlungsoperator `explicit`. Dann wird er vom Compiler nur eingesetzt, wenn im Programm explizit eine Umwandlung programmiert ist (und nicht, wenn eine nötig wäre). `explicit` ist auch bei Konstruktoren sinnvoll, die nicht als Umwandlung verwendet werden sollen: Die automatische Umwandlung `int` in Bruch ist sinnvoll, eine automatische Umwandlung von `double` in einen Vektor `Vec2` wäre mathematisch falsch!

Übungsblatt Nr. 12 Hausübung: Die folgenden Aufgaben sind Pflicht und zählen 1 Punkt!

1) a) Extrahieren Sie aus Hausaufgabe 3 (Multiple-Choice-Test) `SimpleStat.hpp` und inkludieren Sie diese in das C++Programm.
   b) Machen Sie auch hier die Datenattribute `private` und den ganzen Rest `public`. Nehmen Sie zur Demonstration `class` statt `struct`, es wird sich nichts ändern.
   c) Die Initialisierung muss jetzt mit einem Konstruktor erfolgen. Schreiben Sie einen passenden Konstruktor

   ```
   explicit SimpleStat(std::string t) ...,
   ```

   machen Sie die Streamausgabe zum `friend` und `operator+=` zur Methode. Das Programm sollte wieder funktionieren!

2) Klausurvorbereitung STL-Algorithmen:
   a) Speichern Sie in einen `vector<unsigned> x;` 10 Zufallszahlen, die Sie mittels `rand()` generieren.
   b) Geben Sie `x` mit einem `range_for` und dann in umgekehrter Reihenfolge mit `for_each()` aus.
   c) Geben Sie das Minimum und seinen Index in `x` aus.
   d) Geben Sie die Anzahl der Zahlen > 10000 aus.
   e) Geben Sie die Summe der Zahlen aus.
   f) Berechnen Sie das Maximum der Zahlen mit `accumulate`.

3) Klausurvorbereitung Objekte: Die Datei `u3.cpp` enthält ein Programmgerippe, das Sie vervollständigen sollen.

4) Klausurvorbereitung Objekte: Starten Sie mit `u4.cpp` aus dem Übungsordner.

a) Kreieren Sie einen Datentyp `Druckjob` mit den Bestandteilen

```
string name;            // Name der Datei
string user;            // Auftraggeber
int seiten;             // Seitenumfang
bool doppelseitig;      // einseitiger oder doppelseitiger Druck
```

Initialisieren Sie die Bestandteile so, dass doppelsseitig als Defaulteinstellung `true` hat. Schreiben Sie eine Streamausgabe für `Druckjob` analog zu:

```
Druckjob(User: stix, File: "..." Seiten: 13 doppelseitig)
```

Programmieren Sie Konstruktor und Streamausgabe für `Printer`, der einen Papiervorrat von 1000 Blatt haben soll. Die Ausgabe soll so aussehen:

```
Drucker Laserjet:
        Seiten gedruckt: 6600
        Papier verbraucht: 6600
        Papiervorrat: -5600
        Anzall Druckjobs: 450
        Stoerungen: 0
```

b) Der Output zeigt noch einige Schwachstellen: Die Anzahl der Blätter wird falsch berechnet, weil der doppelseitige Druck nicht berücksichtigt wird (Achtung 7 Seiten ergibt 4 Blatt Papier). Der negative Papiervorrat muss auch behoben werden: Testen Sie vor dem virtuellen Ausdruck, ob der Vorrat reicht. Wenn nicht, werfen Sie eine `runtime_exception` der Form

```
Laserjet: kein Papier
```

Schreiben Sie auch einen Destruktor, der die Methode `print_log()` aufruft. Jetzt wird ihr Programm wegen Papiermangel beendet. Fügen Sie in der `submit_job()` Funktion einen `try-catch` Block ein, der die Exception auffängt, Papier nachlegt und den Druckjob erneut dem Drucker übergibt.

c) Bauen Sie noch etwas Zufall ein: Instanzieren Sie einen Zufallszahlengenerator und generieren Sie in `user()` die Seitenanzahl und die Doppelseitigkeit mit

```
uniform_int_distribution<int> seiten{from, to};
bernoulli_distribution ds{p_ds};
```