Ü B U N G E N zu Programmiersprache 1 (Python, C++) WS 2021/2022

Übungsblatt Nr. 8: in der Übungsstunde

1) Definieren Sie den neuen C++-Datentyp `Bruch`, der aus `int z, n` (Zähler und Nenner) besteht.
   a) Erzeugen Sie 4 Brüche `a{1, 2}, b{3}, c{}, d`. Machen Sie die Zuweisung `d = a;`. Setzen Sie den Nenner von `b` auf 7. Kontrollieren Sie im Debugger `gdb` den Inhalt der 4 Objekte nach jeder Operation.
   b) Vergeben Sie Default-Werte für den Bruch: 0 für `z`, 1 für `n`. Testen Sie das im Debugger.
   c) Die Initialisierung auf einen Default-Wert kann man auch als {} schreiben. Der Defaultwert für alle Integertypen und alle Gleitkommatypen ist 0.
   d) Definieren Sie die Funktion `void print(Bruch b)`, die `b` in der Form `(z/n)` ausgibt. Geben Sie alle Objekte aus.

2) a) Programmieren Sie die Bruch-Addition durch eine Funktion
   `Bruch add(Bruch a, Bruch b)`.
   Berechnen Sie damit `a+b` und geben Sie es aus.
   b) Wenn Sie `Bruch` als Rückgabetyp festgelegt haben, brauchen Sie im `return` nur mehr die Initialisierung { ... }.
   c) Programmieren Sie die Ausgabe so, dass die komplette Rechnung gedruckt wird:
   `(1/2) + (3/1) = (7/2)`

3) In dieser Aufgabe wollen wir die 2 Bruchfunktionen so umbauen, dass wir `a + b` für die Addition und `cout << a` für die Ausgabe schreiben können.
   a) Benennen Sie die Funktion `add` um in `operator+` und deren Aufruf `add(a, b)` um in `a+b`.
   b) Gestalten Sie die Ausgabe so, wie Sie sie gerne hätten und "erfreuen" sich an den vielen Fehlermeldungen von C++. Weil C++ keine Ahnung hat, wie es einen Bruch ausgeben soll, listet der Compiler alle ihm bekannten Ausgabeoperatoren auf und sagt ihnen, warum er diese nicht verwenden kann.
   c) Machen Sie aus `print(Bruch)` den `operator<<()` (Stream-Ausgabe). Dieser soll wie üblich in der Form `cout << a` verwendet werden:
   `print(a)` $\Longleftrightarrow$ `cout << a` $\Longleftrightarrow$ `operator<<(cout, a)`
   das erste Argument der Streamausgabe ist also der Stream (`cout` hat den Typ `std::ostream`) und das zweite der Bruch. Naheliegend ist die Signatur
   `void operator<<(ostream os, Bruch b)`. Was sagt C++ dazu?
   d) C++ meckert über etwas `deleted`: Der Grund liegt darin, dass alle Streams nicht kopierbar sind (was C++ durch das Löschen der Kopier-Operation sicherstellt). Unsere Streamausgabe übergibt `cout` aber per Wert (d.h. als Kopie) und deshalb die Fehlermeldung: Übergeben Sie den Stream per Referenz ohne `const`!
   e) Der Typ der Fehlermeldung hat sich geändert :-): invalid operands of void and ... Das liegt an der Verkettung der Sreamausgaben: `cout << a << " + "`
   C++ setzt hier Klammern: `(cout << a) << " + "`. Betrachtet man die zweite Ausgabe so ist ihr erste Argument `cout << a`, d.h. der Rückgabewert der ersten Ausgabe, welcher `void` ist (und nicht `cout`). Jede Streamausgabe sollte daher seinen Ausgabestream als Resultat zurückgeben (wieder als Referenz!).

Übungsblatt Nr. 8 Hausübung: Die folgenden Aufgaben sind Pflicht und zählen 1 Punkt!

1) a) Schreiben Sie die Funktion `int sum_odd(int n)`, die alle ungeraden Integer $\leq n$ aufsummiert und die Summe zurückgibt. Sie müssen das in C++ am besten als `for`-Schleife realisieren, in der Sie diese ungeraden Zahlen durchlaufen und zu einer Summenvariable addieren: `s += i`. Vergessen Sie nicht, diese Summenvariable vorher auf 0 zu setzen.

   b) Schreiben Sie analog zu a) die Funktion `long long factorial(int n)`, die in einer Schleife alle Integer $\leq n$ multipliziert.

   c) Schreiben Sie im Hauptprogramm eine Schleife, mit denen Sie für `i = 2, 4, 6, ... , 30` die Ergebnisse `sum_odd(i)` so ausgeben:
   `sum_odd(10) = 25`

   d) Schreiben Sie im Hauptprogramm eine Schleife, mit denen Sie für `i = 0, 1, 2, ... , 30` die Ergebnisse `factorial(i)` analog ausgeben. Was bemerken Sie?

2) a) Programmieren Sie die Bruchsubtraktion und geben Sie auch die Differenz `a - b` aus.

   b) Geben Sie `-a` aus: Sie erhalten eine Fehlermeldung vom Compiler, weil er diesen Oerator nicht kennt (es ist nicht die Subtraktion, weil diese 2 Operanden hat!) Programmieren Sie auch diesen `operator-(Bruch a)` und geben Sie aus:
   `-a, -b, -(a+b), (-a) + (-b)`

3) a) Programmieren Sie auch Multiplikation und Division von `Brüchen` und geben Sie zusätzlich das Produkt und den Quotienten von `a` und `b` aus. Berechnen Sie auch `(-a) * (-b)`.

   b) Modifizieren Sie die Bruchausgabe so, dass sie bei Nenner 1 nur mehr den Zähler wie einen normalen `int` ausgibt.

4) Kreieren Sie analog zu Bruch den Typ `Vec2` der 2D-Vektoren, die aus `double x, y` bestehen. Programmieren Sie die Addition und Subtraktion sowie die Streamausgabe in der Form `[ x | y ]`. Definieren Sie `A, B, C, D` als Eckpunkte eines Rechtecks mit Seitenlängen 2 und 3 (in beliebiger Lage) und berechnen Sie die Seitenvektoren `AB, BC, CD, DA` und die Diagonalen `AC, BD`. Geben Sie die Vektoren wie folgt aus:
`AB = ...`

Extra-Aufgabe(n): Diese Aufgaben sind freiwillig und zählen 2 Punkte!

5) Lösen Sie die Python-Extraaufgabe von Übung 3, Beispiel 6 (Primzahlen berechnen) mit einem C++-Programm.