Ganze Polynome

Ein ganzes (oder ganzzahliges) Polynom vom Grad n ist eine Funktion der Form $P(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$, mit ganzzahligen Koeffizienten ($a_i \in \mathbb{Z}$) und einem Leitindex a_n ungleich 0. Zur Darstellung von ganzen Polynomen vom Grad n in einem Programm reicht es, die Koeffizienten in einem Vektor mit n+1 Elementen zu speichern. Wir gehen im Weiteren davon aus, dass a_0 an der Indexposition 0 gespeichert ist, a_1 an der Indexposition 1 usw. Implementieren Sie eine Klasse **`Polynom`** zur Repräsentation von ganzen Polynomen mit folgenden Methoden und Operatoren:

- Konstruktor(en): Es wird ein vector<int> mit den Koeffizienten als Parameter übergeben. Es gibt keinen Defaultwert. Der Versuch, einen leeren Vektor zu übergeben, bzw. einen in dem der letzte Eintrag (Leitindex) 0 ist, führt zum Werfen einer Exception vom Typ **`runtime_error`**.
- **`int grad() const`**: Liefert den Grad des Polynoms zurück.
- Operator <<: Gibt das Polynom in einem simplen Format aus: $a_n x\text{^}n + a_{n-1} x\text{^}(n-1) + \cdots + a_1 x\text{^}1 + a_0 x\text{^}0$. Statt der a_i sind die entsprechenden Koeffizienten auszugeben.
  Z.B.: 3x^6+0x^5+-2x^4+0x^3+2x^2+0x^1+-5x^0 für $3x^6 - 2x^4 + 2x^2 - 5$.
- Zusatz für 10 Punkte: **`int wert(int n) const`**: Liefert den Wert des Polynoms an der Stelle n („Einsetzen des Werts **`n`** für x“) zurück.
- Zusatz für 15 Punkte: **`ostream& print_fmt(ostream& o) const`**: Gibt das Polynom in einer angenehmeren Formatierung auf den Stream **`o`** aus und retourniert den Stream **`o`**. Die Exponenten sollen mit dem Zeichen ^ dargestellt werden (z.B.: x^3). Potenzen mit Koeffizient 0 sollen unterdrückt werden (z.B.: x^2+4 statt x^2+0x+4). Negative Koeffizienten sollen ein eventuell davor stehendes Additionszeichen zu einem Subtraktionszeichen ändern (x^2-4 statt x^2+-4), Koeffizienten mit dem Wert 1 sind zu unterdrücken (z.B.: x statt 1x), Exponent 1 wird nicht dargestellt (x statt x^1) und x^0 wird unterdrückt.

Implementieren Sie die Klasse **`Polynom`** mit den notwendigen Konstruktoren, Methoden und Operatoren, sodass jedenfalls das Rahmenprogramm kompiliert und ausgeführt werden kann und die gewünschten Ergebnisse liefert. Achten Sie in Ihren Konstruktoren darauf, dass nur gültige Objekte erstellt werden können. Werfen Sie gegebenenfalls eine Exception vom Typ **`runtime_error`**.

Für Ihr Programm dürfen Sie **nur** die im vorgegebenen Rahmenprogramm angeführten include-Dateien verwenden!

Instanzvariablen sind **`private`** zu definieren und die Verwendung globaler Variablen ist (abgesehen von im Rahmenprogramm eventuell bereits definierten) nicht erlaubt! Die Datenkapselung darf nicht durchbrochen werden. Es ist daher unter anderem nicht erlaubt, Referenzen oder Pointer auf private Instanzvariablen einer Klasse nach außen zu vermitteln, **`friend`**-Deklarationen (mit Ausnahme bei Operatorfunktionen) zu verwenden, oder setter-Methoden zu implementieren, die die Integrität der Daten nicht gewährleisten. Interpretationsspielraum in der Angabe können Sie zu Ihren Gunsten nutzen.

Die Teilaufgaben, bei denen keine Punkteanzahl angegeben ist, gelten als Basisfunktionalität. Für eine positive Beurteilung ist zumindest die Basisfunktionalität zu implementieren. Diese wird mit 30 Punkten bewertet.

Die übrigen Teilaufgaben müssen nicht unbedingt implementiert werden, führen aber im Falle einer korrekten Implementierung zu einer entsprechenden Erhöhung der Punkteanzahl.