Video Games Auswahl

Implementieren Sie die Klassen `Game` und `Selection`:

Ein `Game`-Objekt hat die Instanzvariablen `name` (`string`, nicht leer), `year` (`int`, Erscheinungsjahr zwischen 1981 und 2019, jeweils inklusive), `score` (`int`, Bewertung zwischen 0 und 10, jeweils inklusive) und `platforms` (`vector<Platform>`, nicht leer, Plattformen, auf denen das Spiel verfügbar ist). `Platform` ist eine vordefinierte Enumeration mit den Werten `Platform::Nintendo`, `Platform::Sony`, `Platform::Microsoft` und `Platform::PC`. Für die Klasse `Game` sind folgende Methoden und Funktionen zu implementieren:

- Konstruktor(en) mit 4 Parametern: Name, Erscheinungsjahr, Bewertung und Liste an Plattformen in dieser Reihenfolge. Sollte einer der Parameter nicht die Voraussetzungen erfüllen (z. B. Name ist leer, Jahr bzw. Bewertung nicht im erlaubten Bereich oder Liste der Plattformen ist leer), ist eine Exception vom Typ `runtime_error` zu werfen.
- `bool supports(Platform p) const`: Retourniert `true`, falls das Spiel (`this`-Objekt) auf Plattform `p` verfügbar ist, `false` sonst.
- `int add_platforms(const vector<Platform>& more)`: Fügt alle Plattformen aus `more` zur Liste der Plattformen hinzu, für die das Spiel (`this`-Objekt) verfügbar ist, soweit sie nicht schon in dieser Liste enthalten sind. Zu retournieren ist die Anzahl der neu hinzugefügten Plattformen.
- `operator>`: Vergleicht zwei Spiele. Retourniert `true`, falls der linke Operand "besser" ist als der rechte, `false` sonst. Ein Spiel ist "besser" als ein anderes, wenn es entweder eine höhere Bewertung hat oder dieselbe Bewertung und ein höheres Erscheinungsjahr.
- `static vector<Game> filter(const vector<Game>& games, Platform p)`: Retourniert die Liste der Spiele aus `games`, welche auf der Plattform `p` spielbar sind. Die relative Reihenfolge der Spiele in der retournierten Liste muss jener in `games` entsprechen.
- `operator<<`: Ein Spiel muss in der Form [*name*, *year*, *score*, {*Liste der Plattformen*}] ausgegeben werden, z. B.:
  [Knights of the old Republic, 2003, 6, {Microsoft, PC}]. Der vordefinierte Vektor `platform_names` kann für die Ausgabe der Plattformen verwendet werden.

Ein `Selection`-Objekt hat die Instanzvariablen `name` (`string`, nicht leer), `platform` (`Platform`) und `games` (`vector<Game>`, eventuell leere Spieleliste). Für die Klasse `Selection` sind folgende Methoden und Funktionen zu implementieren:

- Konstruktor mit 2 oder 3 Parametern: Name, Liste von `Game`-Objekten und Plattform in dieser Reihenfolge. Plattform ist optional und per Default `Platform::Nintendo`. Sollte der Name leer sein, ist eine Exception vom Typ `runtime_error` zu werfen.
- `bool ready() const`: Retourniert `false`, wenn die Spieleliste zumindest ein Spiel enthält, das nicht für die Plattform des `this`-Objekts verfügbar ist, `true` sonst.
- `operator<<`: Die Ausgabe eines Objekts vom Typ `Selection` muss in der Form [*name*, *platform*, {*Liste der Spiele*}, `ready`] erfolgen, wobei `ready` dem Text "ready" entspricht sofern die Methode `ready` den Wert `true` zurückliefert, sonst dem Text "not ready" , z. B.:
  [N64, Nintendo, {[Knights of the old Republic, 2003, 6, {Microsoft, PC}], [Mk64, 1996, 9, {Nintendo}]}, not ready].
- Zusatz für 10 Punkte: Erweitern Sie die Klasse `Selection` um die Methode
  `vector<Selection> compare(const vector<Selection>& cmp) const`:
  Ist die Spieleliste des `this`-Objekts leer, so ist eine Exception vom Typ `runtime_error` zu werfen. Andernfalls wird die Liste jener Selektionen aus `cmp` retourniert, die zumindest ein Spiel enthalten, das "besser" ist als alle Spiele in der Spieleliste des `this`-Objekts. Die relative Reihenfolge der Einträge in der retournierten Liste muss jener in der Parameterliste `cmp` entsprechen.
- Zusatz für 15 Punkte: Erweitern Sie die Klasse `Selection` um die Methode `void rearrange()`:
  Die Spiele in der Spieleliste müssen nach Plattform gruppiert werden. Dabei sind die Plattformen in der Reihenfolge, in der sie in der Enumeration definiert sind, zu behandeln. Das heißt, Spiele, die für `Platform::Nintendo` verfügbar sind, müssen in der Liste ganz vorne sein, dann jene Spiele, die für `Platform::Sony` verfügbar sind, gefolgt von Spielen, die für `Platform::Microsoft` verfügbar sind und schließlich die Spiele, die für `Platform::PC` verfügbar sind. Die relative Reihenfolge innerhalb der einzelnen Gruppen muss jener in der ursprünglichen Liste entsprechen. (Beachten Sie bitte, dass die Spiele nur umgeordnet werden sollen. Ein Spiel also nicht in mehrere Gruppen eingetragen wird. Ein Spiel wird z. B. nur dann in die Gruppe `Platform::PC` eingeordnet, wenn es nicht schon zuvor in eine der anderen Gruppen eingeordnet wurde.)

Implementieren Sie die Klassen `Game` und `Selection` 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.