10. Übung des Programmierpraktikums

Abgabetermin: 26. Juni 2017, 23:55 Uhr

Abzugeben sind jeweils die sinnvoll dokumentierten Programmfiles (`*.cpp`, `*.h`) in einem separaten Ordner dessen Name `bsp_`*`nummer`* zu sein hat, d.h., der Ordner `bsp_2` gehört zu Aufgabe 2. **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.**

---

30. Implementieren Sie ein Programm, das ermöglicht, Schach[1] zu spielen. (6+2 Pkt.)

    - Definieren Sie eine abstrakte Klasse `Piece`, die Informationen zu einer Schachfigur speichert. Überlegen Sie, wie Sie die Position der Figur speichern und definieren Sie u.a. eine abstrakte Methode `moveEnabled`, die `true` bzw. `false` zurückgibt, je nachdem, ob der Zug zulässig bzw. nicht zulässig ist. Dabei werden nur die Schachbretträder und die Ziehregeln einzelner Figuren ohne Rücksicht auf die anderen auf dem Schachbrett stehenden Steine berücksichtigt.
    - Leiten Sie von der Klasse `Piece` die einzelnen Schachfiguren ab und implementieren sie jeweils die Methode `moveEnabled`. Beachten Sie dabei die Möglichkeit, dass der Bauer am Anfang zwei Schritte machen darf. (Die Art, wie ein Bauer andere Figuren schlagen kann, und andere Spielregeln mit Bezug auf andere Steine im Spiel lassen Sie außer Acht.)

      Hinweis: Sie können, aber **müssen nicht**, die mehrfache Vererbung verwenden – Details dazu können Sie z.B. unter http://cpp.nope.bz/mehrfach_vererbung.html[2] finden. Vorsicht bei der Basisklasseninitialisierung im Falle von Mehrfachvererbung.
    - Definieren Sie nun eine Klasse `Board`, die zwei Arrays `white` und `black` mit den Einzelnen Figuren beinhaltet. Überladen Sie außerdem den `<<`-Operator, sodass sie eine sinnvolle Ausgabe bekommen (dazu eignen sich z.B. die Zeichen "–", "+" und "|"). Zusätzlich implementieren Sie die Methode `move`, die einen Zug implementiert und ihn nur dann durchführt, wenn er zulässig ist. Benützen Sie dabei die Methode `moveEnabled` und beachten Sie dabei zusätzlich alle Spielregeln **außer**:
      - Rochade,
      - Schlagen "en passant",
      - Verwandeln eines Bauers in eine andere Figurenart,
      - Regeln, die es dem König verbieten, bedrohte Felder zu betreten und Regeln, die es Verbieten, andere Figuren so zu verschieben, dass der König auf einem bedrohten Feld stehen würde und
      - Remis-Regeln.

      Geben Sie `true` bzw. `false` zurück, je nachdem, ob der Zug zulässig bzw. nicht zulässig ist. (Beachten Sie dabei, dass nicht zulässige Züge gar nicht durchgeführt werden, es wird nur `false` zurückgegeben.)

[1] `https://de.wikipedia.org/wiki/Schach`

[2] `http://cpp.nope.bz/mehrfach_vererbung.html`

- Definieren Sie nun eine Klasse `Game`, die ein Schachbrett (Instanz der Klasse `Board`) als Member hat, und ein Spiel implementiert. Dabei werden die Züge über die Tastatur eingelesen.

Zusatzaufg.: Implementieren Sie die Verwandlung eines Bauers in eine andre Figur. (+1 Pkt.)

Zusatzaufg.: Implementieren Sie die Möglichkeit einer Rochade, dabei brauchen Sie **nicht** zu überprüfen, dass der König durch keine feindliche Figur bedroht war, und dass die Felder, die der König benützt, ebenfalls unbedroht sind. (+1 Pkt.)

Zusatzaufg.: Implementieren Sie die en-passant-Regel. (+2 Pkt.)

Zusatzaufg.: Implementieren Sie die Regeln, die es verbieten so einen Zug zu machen, der den König unter Bedrohung bringen würde. Falls Sie die Rochade-Zusatzaufgabe gemacht haben, implementieren Sie auch alle Rochade-Voraussetzungen. (+3 Pkt.)

Zusatzaufg.: Implementieren Sie die Remis-Regeln **außer** der *technischen Remis*. Eine solches Remis tritt dann auf, wenn (+2 Pkt.)

- der am Zug befindliche Spieler keine Zugmöglichkeit hat, sein König sich jedoch nicht im Schach befindet (d.h. wenn der nicht bedrohter allein übrig gebliebener König nicht ziehen kann, ohne bedroht zu werden)
- zum mindestens dritten mal dieselbe Stellung mit demselben Spieler am Zug und denselben Zugmöglichkeiten vorliegt oder
- mindestens 50 Züge lang weder eine Figur geschlagen noch ein Bauer gezogen wurde.

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[3].

Die Verzeichnisnamen mit den Files für die jeweilige Aufgabe **müssen** dem Schema `bsp_nummer` folgen, z.B. enthält das Verzeichnis (der Ordner) `bsp_1` alle Files für Beispiel 1. Andere Verzeichnisnamen zählen als nicht abgegeben.
Keine Lehrzeichen, Sonderzeichen oder Umlaute in File- und Verzeichnisnamen benutzen (Portabilität).

[3] `http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Download/kfu.html`