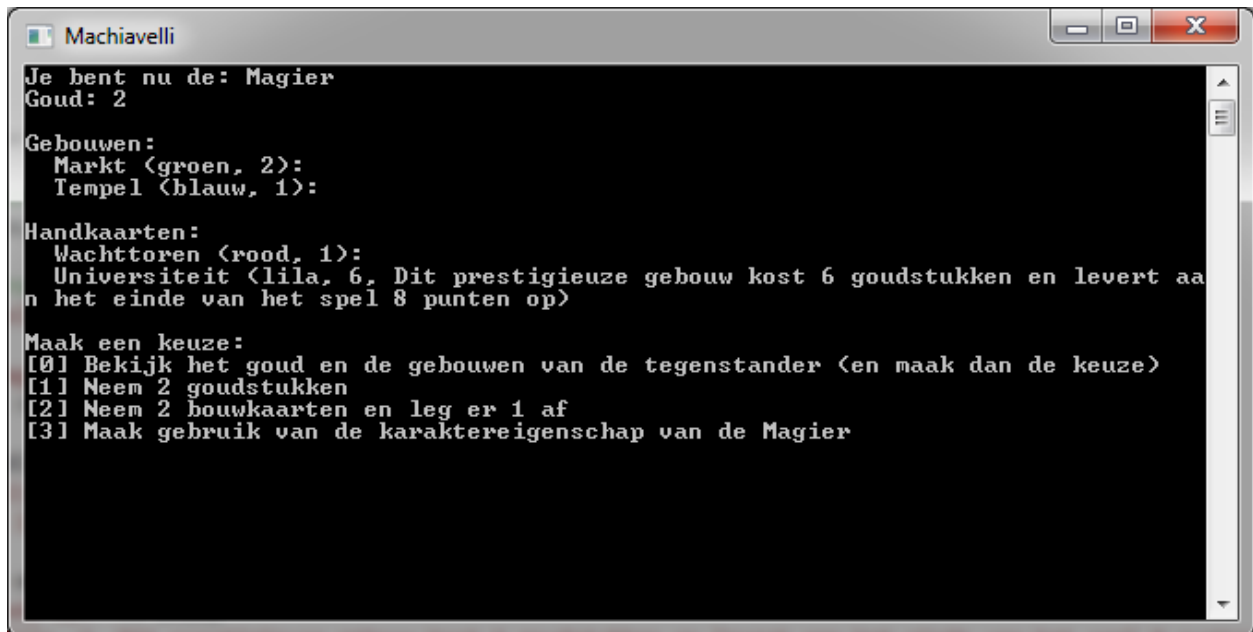


Eindopdracht CPP2

Maak in C++ een applicatie van het spel *Machiavelli*. Zie de bij deze eindopdracht geplaatste pdf voor de spelregels*. Je moet **in tweetallen**** een console applicatie maken die door middel van tekst weergeeft wat jouw gelegde kaarten zijn en/of wat de gelegde kaarten van de tegenspeler zijn. Het is niet verboden om ergens een grafische weergave van te hebben, maar je krijgt hier geen extra punten voor. De console geeft ook aan wat voor keuzes een speler heeft indien deze een keuze moet maken.

Voorbeeld:



```
Je bent nu de: Magier
Goud: 2

Gebouwen:
  Markt <groen, 2>:
  Tempel <blauw, 1>:

Handkaarten:
  Wachtoren <rood, 1>:
  Universiteit <lila, 6, Dit prestigieuze gebouw kost 6 goudstukken en levert aan het einde van het spel 8 punten op>

Maak een keuze:
[0] Bekijk het goud en de gebouwen van de tegenstander <en maak dan de keuze>
[1] Neem 2 goudstukken
[2] Neem 2 bouwkaarten en leg er 1 af
[3] Maak gebruik van de karaktereigenschap van de Magier
```

*Machiavelli is normaalgesproken een spel voor 2 tot 7 spelers, maar voor deze opdracht beperken we ons tot de variant met 2 spelers, of in het geval van groepjes van drie studenten de variant met 3 spelers.

**Er wordt altijd per tweetal gewerkt aan de eindopdracht, tenzij de docent toestemming heeft gegeven om een drietal te vormen (bijvoorbeeld omdat er één student anders alleen zou overblijven). In dit geval worden er door de docent additionele eisen gesteld aan de opdracht.

Gegeven

- Een scan van alle verschillende kaarten is bij deze opdracht gegeven. Deze scans zijn alleen ter kennisneming, de afbeeldingen mogen verder niet gebruikt of verspreid worden (om auteursrechtelijke redenen).
- Voor deze opdracht is een tekstbestand gegeven waarin alle bouwkaarten staan met hun naam, hun waarde, hun kleur, en eventueel hun speciale eigenschap. Sommige kaarten komen meer dan één keer voor in het spel. Kaarten die in het echte spel meerdere keren voorkomen staan ook meerdere keren in dit bestand. Na inlezen heb je dus vanzelf de juiste aantallen van elke kaart.
- Voor deze opdracht is een file gegeven waarin de acht karakterkaarten staan met hun volgordenummer.

- Er is voorbeeldcode gegeven ter ondersteuning van de implementatie van bepaalde eisen. Deze voorbeeldcode mag vrijelijk gebruikt of aangepast worden in de opdracht.
- De koningskaart (de kaart van de kroon die normaalgesproken op een voetstukje staat) hoeft niet als kaart geïmplementeerd te worden. Het enige wat zichtbaar moet zijn is een regeltje tekst waarin aangegeven wordt of de speler in kwestie koning is.
- De overzichtskaart hoeft niet als kaart geïmplementeerd te worden. De tekst op de kaart hoeft alleen voor te komen als "hulptekst". Wanneer er dus voor de hulpoptie gekozen wordt moet de teks van de kaart weergegeven worden.

Eisen

- Het moet in C++ geprogrammeerd zijn
- Er mogen geen memory leaks voorkomen
- Er moet een template class worden gebruikt in de code die *KaartStapel* heet. Deze wordt toegepast voor zowel het stapeltje bouwkaarten als het stapeltje karakterkaarten.
- Als je casts gebruikt mag er *alleen* gebruik worden gemaakt van C++ style casts (static_cast, const_cast, dynamic_cast, en reinterpret_cast).
- Er mag *alleen* gebruik worden gemaakt van smart pointers (unique_ptr, shared_ptr, en weak_ptr). "Gewone" pointers mogen dus niet voorkomen!
- Er moet exception handling plaatsvinden. De applicatie mag *nooit* een exception aan de gebruiker doorgeven, dit moet door middel van een berichtje in de console weergegeven worden, waarna, mocht dit nodig zijn, netjes uit het programma gesprongen wordt.
- Alle code moet *exception safe* zijn. Dat wil zeggen dat het niet mogelijk moet zijn dat memory leaks ontstaan als gevolg van een exception.
- De bouwkaarten en de karakterkaarten moeten beiden door middel van een file stream worden ingelezen.
- Er moet een aparte Machiavelli Client applicatie worden gemaakt en een aparte Machiavelli Server applicatie. Iedere speler start een client op die dan contact maakt met de server. De server houdt de toestand van het spel bij en vraagt de clients een keuze te maken indien nodig. (Er komt voorbeeldcode voor deze netwerkcommunicatie, want dat valt buiten het kader van de cursus.)
- De clients moeten over het netwerk verbinding kunnen maken met de server.
- De server maakt gebruik van een gesynchroniseerde message queue om ervoor te zorgen dat de clients in de juiste volgorde worden afgehandeld. (Ook deze class wordt gegeven.)

Bonuseisen

- De toestand wordt constant opgeslagen d.m.v. file streaming. Wanneer het spel wordt onderbroken (of onverhoopt crasht) moet de toestand weer ingeladen kunnen worden en verder gegaan met het spel.
- De client is multiplatform (d.w.z.: draait op Windows en op Linux of OS X)
- De server is multiplatform

Beoordeling

Indien alle eisen foutloos geïmplementeerd zijn wordt de eindopdracht met een 7 beoordeeld. Alle bonuseisen die correct en volledig geïmplementeerd zijn maken het punt hoger tot maximaal een 10.