Progetto di Programmazione ad Oggetti, a.a. 2020/2021

prof. Francesco Ranzato

1 Scopo

Lo scopo del progetto è lo sviluppo in C++/Qt di una applicazione a libera scelta che soddisfi alcuni vincoli generali obbligatori. Il progetto potrà essere sviluppato da **un singolo studente oppure da una coppia di studenti** e dovrà richiedere approssimativamente 50 ore di lavoro complessivo individuale.

2 Iscrizione al Progetto

I singoli e le coppie andranno preventivamente dichiarati **entro il 13 gennaio 2021**. A tal fine, sarà **obbligatorio** inserire i nominativi dei singoli studenti o delle coppie di studenti nel seguente documento Google Form: https://forms.gle/xVG6EjnYDfcpWS817

3 Vincoli Generali Obbligatori

I **vincoli obbligatori** per il progetto sono i seguenti:

1. Definizione ed utilizzo di una gerarchia G di tipi che soddisfi i seguenti requisiti:

   (a) G include almeno 3 diversi tipi istanziabili

   (b) G include almeno una classe astratta

   (c) G ha altezza ≥ 2

   Ad esempio, in una gerarchia che modella degli oggetti catalogabili in una biblioteca, una classe astratta `LibraryItem` può essere un supertipo di `Libro`, `Rivista`, `Quotidiano`, mentre `Settimanale` e `Mensile` possono essere sottotipi di `Rivista`.

   Per la definizione della gerarchia G usare la fantasia ed ispirarsi ai propri interessi nello scegliere la realtà da modellare mediante i tipi della gerarchia.

2. Definizione ed utilizzo di un opportuno template di classe `C<T>`, con relativi iteratori, i cui oggetti rappresentano un contenitore di oggetti di tipo parametrico `T`. Vi è un unico ovvio requisito da rispettare per il template `C<T>`: non è permesso l'utilizzo dei contenitori STL/Qt (o di altre librerie). Si scelga l'implementazione di `C<T>` ritenuta più opportuna per usare il contenitore come struttura dati della propria applicazione.

3. Definizione di un template di classe `DeepPtr<T>` di puntatori polimorfi "smart" al tipo `T` che implementa una gestione automatica della memoria in modalità "profonda" (deep copy/assignment/destructor) per puntatori polimorfi a `T`, quindi richiedendo che `T` sia un tipo che supporti clonazione e distruzione polimorfa. Si tratta quindi di definire una versione semplificata del tipo `std::unique_ptr<T>` introdotto da C++11 (si veda ad esempio `http://www.cplusplus.com/reference/memory/unique_ptr`).

4. Utilizzo del contenitore `C<T>` per memorizzare puntatori polimorfi smart ai tipi della gerarchia G, ovvero `C` memorizza oggetti di tipo `DeepPtr<B>` dove `B` è qualche tipo base della gerarchia G.

5. Il front-end dell'applicazione deve essere una GUI sviluppata nel framework Qt.

6. **Obbligatorio per progetti in coppia:** Definizione di opportune funzionalità di input/output su file per alcuni dati gestiti dall'applicazione. Non è richiesto l'uso obbligatorio della libreria standard di I/O, è anche possibile valutare l'opportunità di usare le classi per l'I/O fornite da Qt. Per il formato dei file di I/O si possono considerare formati come XML, JSON o altri formati per lo scambio di dati (AXON, YAML, etc.)

3.1 GUI e Design Patterns

La GUI può liberamente trarre ispirazione sia da applicazioni per sistemi desktop che app per sistemi mobile. Si potrà aderire al design pattern **Model-View-Controller** o **Model-View** per la progettazione architetturale della GUI. Qt include un insieme di classi di "view" che usano una architettura "model/view" per gestire la relazione tra i dati logici della GUI ed il modo in cui essi sono presentati all'utente della GUI (si veda `http://qt-project.org/doc/qt-5/model-view-programming.html`). Come noto, la libreria Qt è dotata di una documentazione completa e precisa che sarà la principale guida di riferimento nello sviluppo della GUI, oltre ad offrire l'IDE QtCreator ed il tool QtDesigner.

L'applicazione potrà anche applicare design pattern comunemente utilizzati in C++, ad esempio alcuni pattern utili potrebbero essere (con link alle corrispondenti descrizioni wikipedia):

- Visitor
- Proxy
- Bridge
- Decorator

3.2 Ulteriori Requisiti Obbligatori

Il progetto deve inoltre soddisfare i seguenti requisiti obbligatori:

1. **Separazione tra modello logico e GUI**: in particolare, il modello logico della gerarchia G non ha alcuna relazione con il codice della GUI. La gerarchia G dovrebbe poter essere usata anche da una GUI scritta in un framework diverso da Qt.
2. Deve essere esplicitamente utilizzato del **codice polimorfo** che sfrutti in modo corretto, ragionevole ed estensibile l'organizzazione gerarchica dei tipi di G.

4 Tutorato

La libreria Qt offre una moltitudine di classi e metodi per lo sviluppo di GUI dettagliate e user-friendly. Il tutor Benedetto Cosentino (studente del II anno della Laurea Magistrale) renderà disponibili dei video-tutorato di laboratorio dedicati all'apprendimento delle caratteristiche di base del framework Qt per la progettazione di GUI. Verranno fatti degli annunci su Moodle quando saranno disponibili questi video-tutorato. Ulteriori informazioni al gruppo Facebook **TutorJuniorInformaticaUniPD** e nei gruppi Telegram degli studenti.

5 Valutazione del Progetto

Un buon progetto dovrà essere sviluppato seguendo i principi fondamentali della programmazione orientata agli oggetti, anche per quanto concerne lo sviluppo dell'interfaccia grafica. La valutazione del progetto prenderà in considerazione i seguenti criteri:

1. **Correttezza**: il progetto deve:
   (a) compilare ed eseguire correttamente nella macchina `ssh.studenti.math.unipd.it`, di cui è stata fornita una immagine da usare come macchina virtuale in VirtualBox. **NB: si tratta di una condizione necessaria per la valutazione del progetto.** La compilazione dovrà essere possibile invocando la sequenza di comandi: `qmake -project` $\Rightarrow$ `qmake` $\Rightarrow$ `make`. Se la compilazione del progetto necessita di particolari flag di invocazione per `qmake` (ad esempio `QT += widgets`) oppure un project file (.pro) per `qmake` diverso da quello ottenibile tramite l'invocazione di `qmake -project` allora deve anche essere consegnato un file `progetto.pro` che permetta la generazione automatica tramite `qmake` del `Makefile`, e ciò dovrà essere **esplicitamente** documentato nella relazione.
   (b) soddisfare pienamente e correttamente tutti i vincoli e requisiti obbligatori
2. **Orientazione agli oggetti**: il progetto deve **obbligatoriamente** includere delle chiamate polimorfe. Saranno oggetto di valutazione le seguenti qualità del codice prodotto:
   (a) incapsulamento
   (b) modularità (in particolare, massima separazione tra parte logica e grafica (GUI) del codice)
   (c) estensibilità ed evolvibilità, in particolare mediante polimorfismo

(d) efficienza e robustezza

3. **Funzionalità**: quante e quali funzionalità il progetto rende disponibili, e la loro qualità.
4. **GUI**: utilizzo corretto della libreria Qt; qualità, usabilità e robustezza della GUI.
5. **Relazione**: chiarezza e qualità della relazione, che verterà sui seguenti aspetti:
   (a) descrizione **obbligatoria di tutte le gerarchie di tipi usate**
   (b) descrizione **obbligatoria dell'uso di chiamate polimorfe**
   (c) descrizione **obbligatoria di eventuali formati dei file usati per l'input/output**
   (d) se l'applicazione lo richiede, manuale utente della GUI
   (e) se necessario, indicazione di eventuali istruzioni di compilazione ed esecuzione
   (f) indicazione delle **ore effettivamente richieste** dalle diverse fasi progettuali: analisi preliminare del problema, progettazione modello e GUI, apprendimento libreria Qt, codifica modello e GUI, debugging, testing. In caso di superamento del previsto monte di 50 ore di lavoro complessivo per persona, giustificazione per le ore in eccesso
   (g) **Progetti di coppia:** dovranno essere prodotte **due distinte relazioni individuali** per ogni progetto sviluppato da una coppia di studenti. Ogni studente della coppia dovrà quindi scrivere la propria relazione sull'**intero progetto**. Non saranno accettate relazioni identiche per una coppia di studenti. Inoltre, ogni relazione dovrà contenere una esplicita sezione "Suddivisione del lavoro progettuale" che descriva almeno approssimativamente le diverse responsabilità progettuali della coppia di studenti. Naturalmente la relazione dovrà indicare la coppia di studenti responsabile del progetto consegnato: nome, cognome, numero di matricola.
   (h) **In caso di progetto riconsegnato:** la relazione dovrà **obbligatoriamente** descrivere in modo dettagliato le modifiche apportate al codice rispetto alla precedente versione consegnata, in particolare elencando in modo puntuale le modifiche al codice che risolvono i punti deboli riscontrati nella precedente valutazione del progetto.

Quindi, il progetto dovrà essere **obbligatoriamente** accompagnato da una relazione scritta di **massimo 8 pagine in formato 10pt**. La relazione deve essere presentata come un file PDF di **nome preciso `relazione.pdf`**. La relazione deve anche specificare il sistema operativo di sviluppo e le versioni precise del compilatore e della libreria Qt.

6 Esame Orale e Registrazione Voto

La partecipazione all'esame orale è possibile solo dopo:

1. avere superato con successo (cioè, con voto $\geq$ 18/30) l'esame scritto
2. avere consegnato il progetto entro la scadenza stabilita, che verrà sempre comunicata in Moodle.
3. **ogni studente di un progetto di coppia dovrà fare una consegna distinta del progetto su Moodle**: quindi il codice sarà in comune alla coppia, mentre ogni consegna sarà caratterizzata dalla propria relazione individuale
4. essersi iscritti alla lista Uniweb dell'esame orale

Il giorno dell'esame orale, nel luogo ed all'orario stabiliti, **tipicamente in modalità telematica a meno di diversa comunicazione**, verrà comunicato l'esito della valutazione del progetto assieme ad un sintetico **feedback** sui punti deboli riscontrati nella valutazione del progetto. Il feedback per progetti svolti in coppia naturalmente sarà lo stesso, eventualmente diverso solo per quanto riguarda la relazione individuale. Tre esiti saranno possibili:

(A) Valutazione positiva del progetto con registrazione del voto complessivo proposto **con esenzione dell'esame orale**. Nel caso in cui il voto proposto non sia ritenuto soddisfacente dallo studente, sarà possibile rifiutare il voto oppure richiedere l'esame orale, che potrà portare a variazioni in positivo o negativo del voto proposto.

(B) Valutazione del progetto da completarsi con un **esame orale obbligatorio**. Al termine dell'esame orale, o verrà proposto un voto complessivo sufficiente oppure si dovrà riconsegnare il progetto per un successivo esame orale.

(C) Valutazione negativa del progetto che comporta quindi la **riconsegna del progetto** per un successivo esame orale: in questo caso **il voto dell'esame scritto rimane valido**.

Lo studente che decida di rifiutare il voto finale proposto, con o senza orale, **dovrà riconsegnare il progetto** per un successivo orale (tranne all'ultimo esame orale), cercando quindi di porre rimedio ai punti deboli segnalati nel feedback di valutazione e **descrivendo obbligatoriamente** le modifiche apportare al progetto nella relazione aggiornata. **Il voto sufficiente dell'esame scritto rimane comunque valido**. All'eventuale esame orale lo studente dovrà saper motivare **ogni** scelta progettuale e dovrà dimostrare una **generale conoscenza** del progetto ed una **piena conoscenza** delle parti di propria responsabilità nel progetto svolto in coppia.

7 Regole

7.1 Compilatore, libreria Qt, macchina virtuale

Il progetto deve compilare ed eseguire correttamente sulla **macchina virtuale Linux** per VirtualBox di cui è stata fornita una immagine. La macchina virtuale è dotata di compilatore GNU `g++` 7.3 e di libreria Qt nella versione 5 (5.9.5). È naturalmente possibile sviluppare il progetto su altri sistemi operativi come MacOS/Windows o su altre versioni di Linux/`g++`/Qt. Prima di consegnare il progetto, **è fortemente consigliata** una verifica finale di compilazione/esecuzione/funzionamento del progetto sulla macchina virtuale Linux.

7.2 Cosa consegnare

Tutti i file sorgente `.h` e `.cpp` per il codice C++ assieme al file **`relazione.pdf`** contenente la relazione **individuale**, e ad eventuali file che memorizzano dati necessari per il corretto funzionamento del programma (ad esempio, dei file di input/output necessari al programma). Se la compilazione del progetto necessita di un project file (.pro) per `qmake` diverso da quello ottenibile tramite l'invocazione del comando `qmake -project` allora ciò dovrà essere dichiarato esplicitamente nella relazione e dovrà anche essere consegnato un file `progetto.pro` che permetta la generazione automatica tramite `qmake` del `Makefile`. Nei progetti di coppia, **ogni studente della coppia** dovrà fare una consegna distinta del progetto: quindi il codice sarà in comune alla coppia, mentre ogni consegna sarà caratterizzata dalla propria relazione individuale.

Cosa non consegnare: codice oggetto, eseguibile, file di back-up generati automaticamente da editor o IDE e tutto quanto non necessario per la corretta compilazione ed esecuzione del programma.x

7.3 Come consegnare

All'interno del Moodle del corso, saranno attivati 5 compiti di **Consegna del progetto per l'orale N**, con **N**=1,2,3,4,5 relativi alla consegna del progetto per i 5 esami orali previsti dal corso. Si dovrà consegnare **un unico archivio .zip** contenente **tutti e soli** i file da consegnare. **Attenzione:** La dimensione massima complessiva di tutti i file che verranno consegnati è 256MB (se la dimensione è maggiore il comando di consegna non funzionerà correttamente). **Non saranno accettate altre modalità di consegna** (ad esempio via email).

7.4 Scadenze di consegna

Il progetto dovrà essere consegnato rispettando **tassativamente** le scadenze **ufficiali** (data e ora) previste che verranno rese note tramite il Moodle del corso. Approssimativamente la scadenza sarà circa 10-14 giorni prima dell'esame orale (dipende dal numero di consegne previste per una specifica sessione di consegna).

Per i progetti ritenuti insufficienti, gli studenti della coppia o lo studente singolo dovranno consegnare una nuova versione del progetto per un successivo appello orale.

Prima sessione di esami orali: Le date degli esami orali della sessione regolare con relative scadenze tassative di consegna del progetto sono le seguenti:

Primo orale: mercoledì 10 febbraio 2021, scadenza di consegna: domenica 31 gennaio 2021 ore 23:59

Secondo orale: venerdì 26 febbraio 2021, scadenza di consegna: domenica 14 febbraio 2021 ore 23:59