

ESAME DI PROGRAMMAZIONE E AMMINISTRAZIONE DI SISTEMA

L'esame deve essere svolto singolarmente e deve essere realizzato unicamente con gli strumenti utilizzati nel corso. Dato che i progetti verranno testati con essi, ogni altro strumento potrebbe far fallire, ad esempio, la compilazione e quindi l'esame. In caso di esito negativo dell'esame, lo studente dovrà presentarsi ad un successivo appello d'esame con il progetto previsto per quella sessione.

Controllate spesso il sito del corso per eventuali aggiornamenti!

Questo documento contiene DUE progetti (leggere le note evidenziate):

Progetto C++

- Creazione di un programma a riga di comando con g++, make e doxygen
- Questo progetto deve essere svolto da tutti gli studenti.

Progetto Qt

- Creazione di un programma visuale con le librerie Qt
- Questo progetto deve essere svolto solo dagli studenti dell'insegnamento di "Programmazione e Amministrazione di Sistema" iscritti a partire dall'AA 17/18.
- **Gli studenti di Programmazione e Amministrazione di Sistema degli anni precedenti al 17/18 devono svolgere invece un progetto .NET (CONTATTARE IL DOCENTE).**

Progetto C++ del 17/12/2018

**Data ultima di consegna: entro le 23.59 del
09/12/2018**

**QUESTO PROGETTO E' VALEVOLE COME PROVA PARZIALE DEGLI
INSEGNAMENTI DI Programmazione ad Oggetti C++ (6CFU),
Programmazione C++ (4 CFU), Programmazione e Amministrazione di
Sistema (8 CFU)**

Si richiede la progettazione e realizzazione di una coda FIFO (**Queue**) di elementi generici **T**. In una coda FIFO è possibile inserire e rimuovere un elemento alla volta. L'elemento rimosso è sempre il più vecchio inserito nella coda. Le operazioni d'inserimento e rimozione sono garantite dall'essere eseguite in tempo costante.

A parte i metodi essenziali per la classe (tra cui anche conoscere il numero degli elementi inseriti) devono essere implementate le seguenti funzionalità:

1. Inserimento di un singolo elemento
2. Inserimento di un insieme di elementi presi da una sequenza identificata da due iteratori. L'ordine di inserimento determina l'anzianità dell'elemento.
3. Rimozione dell'elemento più vecchio
4. Accesso in lettura e scrittura all'elemento più recente e all'elemento più vecchio. La sovrascrittura di un elemento non ne cambia l'anzianità.
5. La classe deve includere il supporto agli iteratori (lettura E scrittura) Gli elementi devono essere ritornati seguendo la logica FIFO. La sovrascrittura di un elemento non ne cambia l'anzianità.
6. Deve essere possibile chiedere alla classe se contiene almeno un elemento di un certo valore dato.
7. Scrivete una funzione globale generica `transformif` che data una coda `Q`, un predicato `P`, e un operatore generico `F`, modifica con

l'operatore i valori contenuti nella coda che soddisfano il predicato. In pratica se Q_i è un elemento della coda:

$$Q_i = F(Q_i) \quad \text{se } P(Q_i) == \text{true}$$
$$Q_i = Q_i \quad \text{se } P(Q_i) == \text{false}$$

NECESSARIO: Inserimento e rimozione DEVONO ESSERE IN TEMPO COSTANTE.

Nota 1: Se non indicato diversamente, nella progettazione della classe, è vietato l'uso di librerie esterne e strutture dati container della std library come `std::vector`, `std::list` e simili. E' consentito il loro uso nel codice di test nel main.

Nota 2: Non potete utilizzare i costrutti C++11 e oltre.

Nota 3: Nella classe, è consentito l'uso della gerarchia di eccezioni standard, delle asserzioni e della gerarchia degli stream.

Nota 4: Per vostra sicurezza, tutti i metodi dell'interfaccia pubblica che implementate devono essere esplicitamente testati nel main.

Nota 5: Non dimenticate di usare Valgrind per testare problemi di memoria

Nota 6: Evitate di usare "test" come nome dell'eseguibile. Potrebbe dare dei problemi sotto msys.

Alcune note sulla valutazione del Progetto C++

- Se in seguito a dei test effettuati dai docenti in fase di valutazione (es. chiamate a funzioni non testate da voi), il codice non compila, l'esame NON è superato.
- Implementazione di codice non richiesto non dà punti aggiuntivi ma se non corretto penalizza il voto finale.
- Gli errori riguardanti la gestione della memoria sono considerati GRAVI.
- La valutazione del progetto non dipende dalla quantità del codice scritto.
- NON usate funzionalità C di basso livello come `memcpy`, `printf`, `FILE` ecc... Se c'è una alternativa C++ DOVETE usare quella.
- NON chiedete ai docenti se una VOSTRA scelta implementativa va bene o meno. Fa parte della valutazione del progetto.
- PRIMA DI SCRIVERE CODICE LEGGETE ACCURATAMENTE TUTTO IL TESTO DEL PROGETTO.

Progetto Qt del 17/12/2018

**Data ultima di consegna: entro le 23.59 del
09/12/2018**

GLI STUDENTI ISCRITTI A PROGRAMMAZIONE E AMMINISTARZIONE DI SISTEMA A PARTIRE DALL'AA 17/18 DEVONO SVOLGERE ANCHE QUESTO PROGETTO.

Il progetto richiede la creazione di un tool, provvisto di interfaccia grafica, per ottenere una serie di informazioni relative a un file di testo. Nello specifico deve poter calcolare e visualizzare il numero di:

- Caratteri (inclusa punteggiatura e spazi)
- Caratteri (spazi esclusi)
- Parole
- Frasi - si consideri che le frasi terminino con [., ! o ?]
- Paragrafi - si consideri che i paragrafi terminino con un a capo

Il tool deve quindi consentire di:

1. Caricare un file di testo e visualizzarlo
2. Stimare correttamente e visualizzare le informazioni sopra elencate, cioè: numero di caratteri, caratteri (spazi esclusi), parole, frasi e paragrafi.

I file da dover esaminare utilizzando il tool sono quelli presenti in allegato: *Il_lupo_e_il_cane.txt* e *La_volpe_e_luva.txt*.

Nota 1: Utilizzate la versione 5.6 della libreria Qt.

Alcune note sulla valutazione del Progetto Qt

- Se in seguito a dei test effettuati dai docenti in fase di valutazione (es. chiamate a funzioni non testate da voi), il codice non compila, l'esame NON è superato.
- Implementazione di codice non richiesto non dà punti aggiuntivi ma se non corretto penalizza il voto finale.
- NON verrà valutata l'efficienza dell'applicativo sviluppato.
- Gli errori riguardanti la gestione della memoria sono considerati GRAVI.
- La valutazione del progetto non dipende dalla quantità del codice scritto.
- NON chiedete ai docenti se una VOSTRA scelta implementativa o la configurazione dell'interfaccia grafica va bene o meno. Fa parte della valutazione del progetto.
- NON chiedete ai docenti come installare QtCreator e le librerie Qt
- PRIMA DI SCRIVERE CODICE LEGGETE ACCURATAMENTE TUTTO IL TESTO DEL PROGETTO.

Consegna

La consegna del/dei progetti è costituita da un archivio .tar.gz avente come nome la matricola dello studente. L'archivio deve contenere una e solo una cartella con lo stesso nome dell'archivio (senza estensione .tar.gz). Nella root della cartella devono essere presenti:

1. Un **makefile** (per poter compilare il progetto DA RIGA DI COMANDO) che deve compilare tutto il progetto chiamato "Makefile" (attenzione alle maiuscole). Se la compilazione fallisce, il progetto non viene considerato.
2. Tutti i **sorgenti** (commentati come avete visto ad esercitazione) del progetto e organizzati a vostro piacimento.
3. Il file di **configurazione di Doxygen** per la generazione della documentazione chiamato "Doxyfile" modificato per generare documentazione HTML. **GMail ha bloccato l'invio di file con estensione .js quindi non è più possibile allegare la documentazione in formato html.**
4. **Relazione in PDF** con descrizione del progetto contenente informazioni relative al design e/o analisi del progetto. La relazione serve per capire il perchè delle vostre scelte nell'implementazione o di design. Nella relazione mettere anche Nome, Cognome, Matricola ed E-Mail.
5. **Chi deve consegnare anche il "Progetto Qt", metta tutti i file sorgenti corrispondenti in una sotto-cartella "Qt".**
6. L'archivio NON deve contenere file di codice oggetto, eseguibili etc..

L'eseguibile che verrà prodotto non deve richiedere alcun intervento esterno (es. input da tastiera).

Per creare l'archivio è sufficiente lanciare il comando di msys:

- `tar -cvzf 123456.tar.gz 123456`

dove "123456" è la directory che contiene tutti i file da consegnare.

Ad esempio una struttura dell'archivio può essere questa:

```
123456
|--main.cpp
|--project.h
|--Doxyfile
|--...
|--Qt (SOLO PER PROGETTO Qt)
|  |--*.pro
|  |--MainWindow.cpp
|  |--Main.cpp
|  |--MainWindow.ui
|  |--dataset
|     |--panda
|     |--saxophone
|     |--...
|--...
```

Indirizzo Mail: **corsocpp.ciocca@gmail.com**

Mettere come tag all'oggetto della mail:

[c++-consegna] Per consegnare ufficialmente il progetto
Potete fare più consegne e solo l'ultima verrà ritenuta valida.

[c++-test] Per testare il meccanismo di consegna
Verrà eseguita una compilazione del **Progetto C++** (differita e con cadenza oraria 0.00, 1.00, 2.00,...) e restituita una mail con l'esito. Potete fare tutti i test che volete. E' vivamente consigliato effettuare almeno una prova di compilazione (i progetti che non compilano, non vengono considerati). Il Progetto Qt non verrà compilato.

NOTA: questa mail deve essere usata esclusivamente per la consegna dei progetti. Per ogni altra questione usare le mail dei docenti.