

# ESAME DI PROGRAMMAZIONE C++ (8 CFU)

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 anche 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 CONTATTARE IL DOCENTE.**

# Progetto C++ del 26/01/2022

**Data ultima di consegna: entro le 23.59 del  
15/01/2022**

**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), "Programmazione C++" (8CFU)**

Il progetto richiede la progettazione e realizzazione di una classe che implementa una matrice sparsa (**SparseMatrix**) di elementi generici **T**. Una matrice sparsa è una matrice nella quale solo gli elementi esplicitamente inseriti sono fisicamente memorizzati. Dal punto di vista logico la matrice sparsa si deve comportare come una normale matrice: alla richiesta dell'elemento in posizione (i,j) deve essere ritornato l'elemento inserito oppure un valore di default (D). Questo valore di default è definito dall'utente e deve essere passato in fase di costruzione della matrice.

A parte i metodi essenziali per la classe (tra cui conoscere il numero di righe/colonne, numero degli elementi inseriti, il valore dell'elemento D, ecc...), devono essere implementate le seguenti funzionalità:

1. La matrice deve essere realizzata in modo tale che siano esplicitamente memorizzati solo gli elementi inseriti dall'utente (utilizzo di memoria minimale). In pratica se la dimensione logica della matrice è NxM, non bisogna memorizzare tutti gli NxM elementi ma solo quelli effettivamente inseriti dall'utente.
2. La classe deve includere il supporto agli iteratori (in sola lettura) di tipo forward per l'accesso ai soli elementi inseriti nella matrice. L'iteratore deve ritornare un oggetto `element` che contiene le coordinate (i,j) e il valore inserito in quella posizione. I valori in `element` non devono essere modificabili. Non è importante l'ordine con cui sono ritornati gli elementi.
3. L'inserimento di un valore alle coordinate (i,j) deve avvenire con un metodo `set`. Se l'utente inserisce esplicitamente un valore di default D, questo è inserito normalmente. Se alle coordinate (i,j) esiste già un valore, questo viene sovrascritto.
4. La lettura di un valore alle coordinate (i,j) deve avvenire tramite `operator()`. Se alle coordinate (i,j) non c'è un valore esplicito, deve essere ritornato il valore di default D.

5. Scrivete una funzione globale generica `evaluate` che data una matrice sparsa `M` e un predicato `P`, ritorna quanti valori in `M` (compresi i default) soddisfano `P`. Testate diversi usi della funzione anche su diversi tipi di dati custom.

Utilizzare dove opportuno la gestione degli errori tramite asserzioni o eccezioni.

**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:** A parte `nullptr`, non potete utilizzare altri costrutti C++11 e oltre.

**Nota 3:** Nella classe, è consentito l'uso della gerarchia di eccezioni standard, delle asserzioni, la gerarchia degli stream e la funzione `std::swap`.

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

**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 26/01/2022

**Data ultima di consegna: entro le 23.59 del  
15/01/2022**

**QUESTO PROGETTO E' VALEVOLE COME PROVA PARZIALE DELL'INSEGNAMENTO DI "PROGRAMMAZIONE C++" (8CFU) GLI STUDENTI ISCRITTI A PROGRAMMAZIONE E AMMINISTRAZIONE DI SISTEMA A PARTIRE DALL'AA 17/18 DEVONO SVOLGERE ANCHE QUESTO PROGETTO.**

Il progetto richiede la creazione di un **editor di testo**. Le funzionalità richieste sono le seguenti:

- Creazione di un nuovo file vuoto *Nuovo*;
- Apertura e visualizzazione di un file di testo con estensione ".txt", funzione *Apri*;
- Possibilità di modificare il testo in corrispondenza del cursore (cancellazione, inserimento);
- Salvataggio del file, *Salva* per sovrascrivere la versione precedente del file o *Salva con nome...* per specificare il nome di un nuovo file;
- *Ricerca* di una stringa (no regular expressions) all'interno del testo. Tale funzionalità prevede l'apertura di una dialog come la seguente:



*Match case* impone che il match tra la stringa e il testo sia esatto (maiuscole e minuscole)

Tutte i *match* nel testo devono essere evidenziati, se invece la stringa non è trovata, deve essere visualizzato in basso alla dialog il seguente messaggio in rosso "Nessuna occorrenza trovata".

**Nota:** Utilizzare la versione 5.12 o superiori 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
|  |--...
```

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.

**NOTA:** Se ricevete degli errori su nullptr, dovete modificare le opzioni di compilazione nel make aggiungendo l'attivazione delle funzionalità c++11. Nei comandi di compilazione aggiungete l'opzione:

-std=c++0x