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.

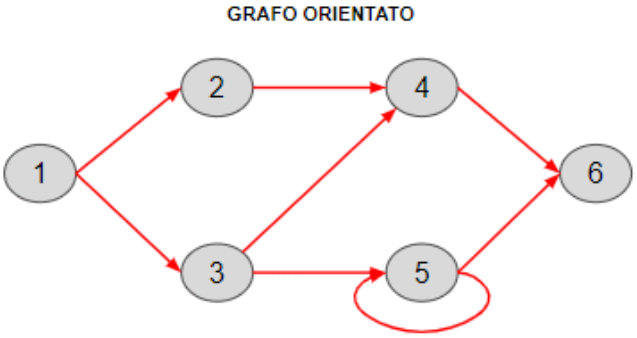
Progetto Qt

- 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 28/02/2024

Data ultima di consegna: entro le 23.59 del 16/02/2024

Il progetto richiede la progettazione e realizzazione di una classe generica che implementa un **grafo orientato**. Un grafo è costituito da un insieme di nodi e archi. I nodi sono rappresentati da un generico identificativo **T** (es. un numero, una stringa, un oggetto, ecc...). Gli archi mettono in relazione due nodi creando un collegamento tra loro. Un esempio di grafo i cui nodi sono identificati da numeri, è mostrato in figura:



nodi destinazione

nodi di partenza	1	2	3	4	5	6
1	0	1	1	0	0	0
2	0	0	0	1	0	0
3	0	0	0	1	1	0
4	0	0	0	0	0	1
5	0	0	0	0	1	1
6	0	0	0	0	0	0

Requisito essenziale del progetto è che il grafo deve essere implementato mediante matrici di adiacenza di booleani come in figura. Non possono essere usate liste.

A parte i metodi essenziali per la classe (tra cui conoscere il numero di nodi e archi), devono essere implementate le seguenti funzionalità:

1. La classe deve includere il supporto al solo `const_iterator` di tipo forward sui nodi. L'iteratore itera sull'insieme degli identificativi dei nodi contenuti nel grafo e l'ordine con cui vengono ritornati non è rilevante.

2. Deve essere possibile aggiungere e rimuovere un nodo usando il suo identificativo. Quando si aggiunge o si rimuove un nodo, bisogna aggiornare la matrice di adiacenza. Gestire i casi: nodo da aggiungere già esistente, nodo da eliminare non esistente.

3. Deve essere possibile aggiungere e rimuovere un arco usando la coppia di identificativi dei nodi che collega. Quando si aggiunge o si rimuove un arco, bisogna aggiornare la matrice di adiacenza. Gestire i casi: arco non esistente e arco già esistente.

4. Deve essere possibile interrogare il grafo per sapere se esite un nodo tramite un metodo `existsNode` e se una coppia di nodi è connessa da un

arco tramite un metodo `existsEdge`. Nell'esempio in figura con I nodi identificati da numeri, existsNode(9)=false, existsEdge(1,2)=true, existsEdge(4,3)=false.

Possono essere trascurate considerazioni di efficienza di accesso ai dati e di occupazione di memoria.

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 se non indicato diversamente.

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. Evitate di fare dei test interattivi. Fato solo test automatici.

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. Usate sempre il nome **main.exe**.

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'e' 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.

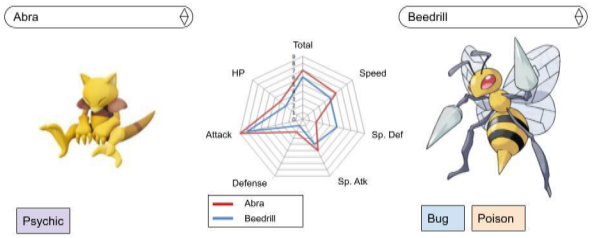
Progetto Qt del 28/02/2024

Data ultima di consegna: entro le 23.59 del 16/02/2024

Il progetto d'esame richiede la progettazione e realizzazione di un'interfaccia grafica per la visualizzazione di un pokédex.

Name	Type	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed
Bulbasaur	GRASS POISON	318	45	49	49	65	65	45
Ivysaur	GRASS POISON	405	60	62	63	80	80	60
Venusaur	GRASS POISON	525	80	82	83	100	100	80
Venusaur Mega Venusaur	GRASS POISON	625	80	100	123	122	120	80
Charmander	FIRE	309	39	52	43	60	50	65
Charmeleon	FIRE	405	58	64	58	80	65	80
Charizard	FIRE FLYING	534	78	84	78	109	85	100
Charizard Mega Charizard X	FIRE DRAGON	634	78	130	111	130	85	100

Esempio finestra principale



Esempio finestra di confronto

L'interfaccia sarà costituita da una finestra principale in cui visualizzare l'elenco dei pokemon con le informazioni annesse a ciascuno di essi (immagine di esempio a sinistra). Una seconda finestra in cui dovrà essere possibile confrontare le caratteristiche tra due pokémon (esempio nell'immagine a sinistra). Per l'elenco dovranno essere implementate le seguenti funzionalità:

1. Visualizzazione delle informazioni per tutti i 1190 pokémon (contenute nel file *pokedex.csv*). Per ciascuna riga riportare le informazioni relative a ciascuno (nome, tipo, attacco, ecc.);
2. Inclusione/esclusione dall'elenco in base alla tipologia di pokémon selezionata/deselezionata;
3. Ordinamento dell'elenco in base a una delle colonne della tabella.

Per la finestra di confronto sono richieste le seguenti funzionalità:

1. Selezione dei due pokémon da confrontare (non deve essere consentito confrontare un pokémon con se stesso);
2. Visualizzazione dell'immagine raffigurante ciascun pokémon (immagini presenti nella cartella *images*) e della tipologia di pokémon.
3. Creazione e visualizzazione di un grafico (a scelta dello studente) in cui riportare il confronto tra i due pokémon.

Nota 1: Utilizzare preferibilmente la versione 5.12.11 della libreria Qt (la stessa installata sulla VM).

Nota 2: Si renda il contenuto dell'applicazione adattivo rispetto alla dimensione della finestra.

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. Fà 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 avviene tramite la piattaforma di eLearning ed 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.
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 o console Linux):

- 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
     |--…
```