Progetto "C++"

valevole come esame "parziale" dei corsi

Programmazione ad Oggetti C++ (6CFU)
Programmazione C++ (4 CFU)
Programmazione e Amministrazione di Sistema (8 CFU)

Il progetto deve essere svolto singolarmente e deve essere realizzato unicamente con gli strumenti utilizzati nel corso. Dato che il progetto verrà testato 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.

Il progetto deve essere contenuto in 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). La cartella deve contenere:

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 "doxygen.conf" modificato per generare documentazione HTML. **Cartella doc con la documentazione HTML generata.**
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. L'archivio dovrà contenere solo ciò che è specificato nei punti (1-4). 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.

Consegna del progetto

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 (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).

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

Alcune note sulla valutazione del progetto

- 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. Fà parte della valutazione del progetto.
- PRIMA DI SCRIVERE CODICE LEGGETE ACCURATAMENTE TUTTO IL TESTO DEL PROGETTO.

Alla data dell'esame, per ogni studente verrà discusso il progetto.

Controllate spesso il sito del corso per eventuali aggiornamenti!

GLI STUDENTI CHE HANNO NEL PIANO DI STUDI IL CORSO DI C++ DA 6 CREDITI (Programmazione ad Oggetti C++) DEVONO FARE UNA INTEGRAZIONE.

Scegliere dall'elenco sottostante 2 o 3 argomenti sui quali produrre una relazione e una presentazione (tipo lezione con esempi d'uso e commenti):

1. Strutture dati della Standard Template Library: (vector, queue, map, set, stream, ...). Ogni struttura dati è un argomento.
2. Pattern PIMPL (o Handle Class)
3. Ridefinizione e overloading degli operatori new e delete (locale, globale, placement new, ...)
4. Implementazione del polimorfismo (VTABLE, ...)
5. Metaprogrammazione con i template

Partite dai libri del corso con particolare attenzione ai libri "Thinking in C++".

CONSEGNA: Il materiale va consegnato INSIEME AL PROGETTO, indicando nei documenti a quale componente del gruppo si riferisce l'integrazione.

NOTA 1: A partire dall'appello di Febbraio/09, TUTTI gli studenti da 6 crediti (AA 08/09 e precedenti) DOVRANNO portare il programma attuale del corso E l'integrazione.

NOTA 2: Chi non ha superato l'esame ma ha fatto l'integrazione in modo accettabile, può ripresentarla tale e quale con il successivo progetto d'esame.

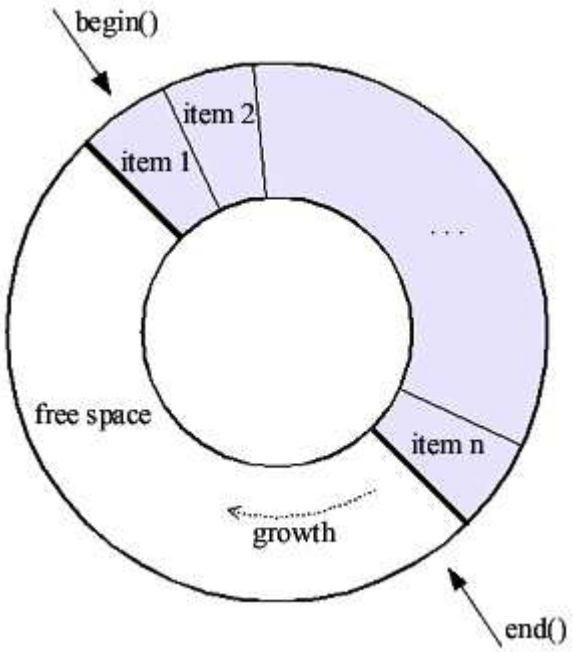
NOTA 3: Chi non ha fatto una integrazione accettabile, non avrà verbalizzato il voto fino a quando non consegnerà una integrazione migliore. In questo caso la consegna andrà comunicata anche all'indirizzo mail del docente.

Per ulteriori chiarimenti contattare il docente.

Progetto d'esame del 15/02/2016

Data ultima di consegna: entro le 23.59 del 08/02/2016

1) Il progetto richiede la realizzazione di una classe **cbuffer** generica che implementa un buffer circolare di elementi di tipo **T** (vedi figura seguente).



Il buffer ha una capacità fissa, decisa a costruzione. L'inserimento accoda gli elementi finché il buffer non è pieno. Una volta riempito, i nuovi dati vengono scritti partendo dall'inizio del buffer, sovrascrivendo i vecchi.

La classe, oltre ai metodi di uso comune (metodi fondamentali, capacità, numero elementi inseriti, etc), deve permettere:

- inserimento di un nuovo elemento (in coda)
- cancellazione di un elemento (in testa)
- accesso in lettura e scrittura dell'elemento i-esimo tramite operatore [].
- supporto agli iteratori

L'inizio del buffer circolare coincide con l'elemento più vecchio che è stato inserito. Tutti i metodi di accesso (operatori, iteratori, etc.) devono tenere in considerazione questa convenzione:

cbuffer[0] è l'elemeno più vecchio
cbuffer.begin() è l'iteratore all'elemento più vecchio

2) Scrivere una funzione globale check che, dati un cbuffer generico **cb** e un predicato unario generico **P** (cioè che prende un solo valore), per ogni elemento cb[i] contenuto nel cbuffer, stampa a console:

```
"[i]: true"  quando P(cb[i]) è vero        oppure

"[i]: false" quando P(cb[i]) è falso
```

con i l'indice dell'elemento.

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

Nota 3: Per vostra sicurezza, tutti i metodi dell'interfaccia pubblica che implementate devono essere esplicitamente testati nel main.

Nota 4: Non dimenticate di usare Valgrind per testare problemi di memoria

Nota 5: Evitate di usare "test" come nome dell'eseguibile. Potrebbe dare dei problemi sotto msys.