Programmazione

PROVA SCRITTA – 13 Gennaio 2021
Parte I – LINGUAGGIO C++ e OOP

1 (12 punti)

Si descriva il funzionamento del polimorfismo a tempo di esecuzione (runtime) in C++ (8 punti)
Un costruttore può essere virtuale (motivare la risposta) ? Un metodo virtuale può essere statico (motivare la risposta) ? (4 punti)

2 (8 punti)

Si descrivano le funzioni costruttore nelle classi C++, indicandone in particolare la sintassi, le modalità di inizializzazione delle variabili membro della classe e come vengono chiamati i costruttori all'interno di una gerarchia di classi. Descrivere il funzionamento del costruttore di copia di default e quando è necessario definirne uno specifico.

3 (10 punti)

Descrivere il meccanismo delle eccezioni nel C++: come si dichiarano, lanciano, prendono (anche in relazione all'ereditarietà) (8 punti). Spiegare le maggiori differenze tra uso di eccezioni e l'uso di valori di ritorno per indicare la presenza di errori a *runtime*. (2 punti)

Programmazione

PROVA SCRITTA – 13 Gennaio 2021
Parte II – PROGRAMMAZIONE

4 (12 punti)

Si implementi la classe `MailMessage` che rappresenta una mail con titolo, mittente, destinatario e testo, e booleano che indica che la mail è stata letta. Si implementi la classe `MailInbox` che rappresenta una casella di ricezione di e-mail, con funzioni per aggiungere e-mail e leggerle (per es. data la posizione).
Si implementi una classe `MailIcon` che disegna l'icona del programma di posta e il numero di e-mail non lette. Si implementi la classe `MailNotifier` che scrive il titolo dell'ultima mail ricevuta. Si usi il design pattern Observer per l'implementazione di `MailIcon` e `MailNotifier`.

5 (12 punti)

Si disegni e implementi una classe `NotePad` che rappresenta un blocco note composto da annotazioni specificate da un titolo; non è possibile avere più di un'annotazione con un serto titolo. Ogni annotazione (classe `Note`) è composta da un titolo, un testo ed una collezione di allegati. Gli allegati possono essere sia immagini che video o file audio (classi `Image`, `Video`, `Audio`) che estendono una comune classe base (`MultimediaDocument`) che fornisce un metodo `show()` che mostra l'oggetto (caratterizzato da *path* del file multimediale, dimensione - per immagini e video - e durata - per video e audio). La classe `Note` ha a sua volta un metodo `show()` che mostra in ordine il titolo, testo e la collezione di allegati nel loro ordine di inserimento nella nota.
Deve essere possibile inserire nel `NotePad` una nuova nota (metodo `add`), cancellare una nota con un certo titolo (metodo `remove`) o mostrare la nota con un certo titolo (metodo `show`). Per implementare la classe `NotePad` si consiglia l'uso di `template<class key_type, class mapped_type> std::map`, di cui si riportano i seguenti metodi utili allo sviluppo delle classi:

Inserimento e selezione di elementi:
```
mapped_type& operator[] (const key_type& k);
```

Ricerca di elementi (ritornano un iteratore a `map::end()` nel caso l'elemento cercato non sia presente):
```
iterator find (const key_type& k);
const_iterator find (const key_type& k) const;
```

Cancellazione di elementi:
```
void erase (iterator position);
size_type erase (const key_type& k);
void erase (iterator first, iterator last);
```

Si ricorda che gli iteratori di una mappa puntano agli elementi che sono di tipo `std::pair<const key_type, mapped_type>`, e che usando i membri `first` e `second` si ottengono la chiave e il valore cui puntano.
Si consiglia l'uso di `list` o `vector` per le collezioni di allegati.

6 (12 punti)

Definire una classe `Histogram` che rappresenti un istogramma. Il costruttore deve consentire di impostare il numero degli elementi (classi) dell'istogramma (2 punti). Deve essere possibile impostare e ottenere la quantità associata ad ogni elemento dell'istogramma, si deve poter impostare il formato di stampa per avere il valore assoluto degli elementi o quello relativo (8 punti) e lo si deve poter visualizzare (2 punti).