Corso di Laboratorio di Programmazione

Assegnamento 1 – Immagini
27/11/2019

Definire la classe Image che rappresenta un buffer contenente un'immagine a toni di grigio. L'immagine è composta da una matrice di larghezza W e altezza H, i cui elementi sono i pixel dell'immagine. Ogni pixel è rappresentato da un unsigned char. Poiché il numero di pixel può essere molto grande, essi devono essere scritti su un blocco di memoria allocato dinamicamente, avente dimensione WxH.

La classe deve avere le seguenti funzioni mebro:

- Un costruttore di default, che crea un oggetto Image senza buffer (con opportuna inizializzazione delle variabili membro);
- un costruttore che accetta le dimensioni W e H, e che inizializza un oggetto opportunamente, inizializzando la memoria allocata a 0;
- se ritenuto necessario, un costruttore di copia e l'operatore assegnamento di copia (devono essere effettuate deep copy);
- un costruttore di spostamento e l'operatore assegnamento di spostamento;
- una funzione at() che accetta due argomenti di tipo int e ritorna una reference al pixel di coordinate W e H;
- una corrispondente funzione at() const che accetta gli stessi parametri – che tipo deve essere restituito in questo caso?
- una coppia di funzione safe_at() e safe_at() const, analoghe alle precedenti, che accedono al buffer previa verifica;
- l'overloading di operator<<, che stampa a schermo tutti i valori dei pixel contenuti nell'immagine.

Deve essere inoltre implementato ogni altro metodo ritenuto necessario (distruttore?) per evitare memory leak. **I memory leak comportano una forte decurtazione del punteggio**, perciò si consiglia di **testare bene** il funzionamento del codice.

Il software deve essere organizzato opportunamente nei seguenti file:

- Image.h: header contenente la dichiarazione della classe ed eventuali funzioni definite in-class;
- Image.cpp: file contenente la definizione delle funzioni membro;
- ImageProcessing.cpp: file contenente il main che testa la classe Image.

Questi file devono essere contenuti in un'unica directory chiamata ImageLib, che deve essere compressa in un archivio .zip e caricata su elearning.

Non sono date specifiche riguardo al file ImageProcessing.cpp: è possibile testare la classe a piacere. Si fa presente che in fase di correzione la classe sarà testata con un ImageProcessing.cpp diverso, che verificherà l'eventuale presenza di errori o debolezze del codice.

La suddivisione del codice nei vari file deve essere effettuata seguendo le linee guida viste a lezione. Errori in questo passaggio (per esempio, includere un file .cpp) comportano una decurtazione del punteggio.

Precisazioni

- Il buffer citato deve essere allocato usando gli operatori new e delete, non tramite contenitori STL o di altre librerie;
- Il buffer è un singolo array, non un insieme di array (per esempio, un array separato per ogni riga dell'immagine);
- Il codice sviluppato deve essere robusto, quindi gli input di ogni funzione devono essere verificati, e le funzioni membro devono sempre rispettare gli invarianti;
- Nella classe, inserite un commento che indica quali sono gli invarianti che la vostra classe deve rispettare;
- Il formato di stampa è il seguente: i valori appartenenti a una singola riga dell'immagine sono stampati di seguito, separati da uno spazio. Alla fine della riga si inserisce un a capo;
- Nel buffer devono essere inseriti numeri, non caratteri. Come detto a suo tempo a lezione: char è il tipo per rappresentare i caratteri, mentre unsigned char e signed char sono tipi usati per rappresentare interi di 8 bit. Quindi l'inizializzazione del buffer deve essere fatta a 0, non a '0' (codice ASCII corrispondente al carattere '0');
- Per stampare un unsigned char come valore numerico è necessario castarlo: std::cout << static_cast<unsigned int> (my_variable_name).