Corso di Laurea in Fisica
Esame di Laboratorio II (I modulo)

01/03/2018

Istruzioni

Si risolvano i seguenti esercizi, scrivendo codici in C++ o Macro di ROOT. Ai fini della valutazione, il primo criterio che deve essere soddisfatto è che il codice compili senza errori ed esegua realizzando le funzionalità richieste dal testo. Per la valutazione sarà inoltre tenuto in considerazione il fatto che i codici siano scritti con ordine, utilizzando opportunamente l'**indentazione** e i **commenti**. Si richiede infine di iniziare i codici con una riga di commento contenente il comando necessario per creare l'eseguibile o per lanciare la Macro di ROOT.

1 Implementazione di un programma frazione

Utilizzando il file `frazione.h`, implementare una classe che rappresenti una frazione aritmetica. Gli oggetti di questa classe avranno due attributi `private`: il numeratore e il denominatore, ovviamente espressi come numeri interi. L'implementazione deve prevedere:

1. Un default constructor che inizializzi la frazione a numeratore 0 e denominatore 1.
2. Un constructor che riceva come input il numeratore e il denominatore della frazione.
3. Un copy constructor, che crei una copia della `frazione` a partire da un oggetto `frazione` già esistente.
4. L'overload degli operatori + (somma), - (sottrazione), * (moltiplicazione), / (divisione), = (uguaglianza).
5. Il destructor, che deve disallocare tutta la memoria utilizzata dall'oggetto `frazione` (potrebbe anche essere che il distruttore non debba eseguire nessuna operazione specifica e quindi possa essere un metodo vuoto).
6. Il metodo `compara` che permette di confrontare l'oggetto frazione su cui viene richiamato con un altro oggetto `frazione` passato come parametro. Questo metodo deve restituire 0 se le due frazioni sono uguali, -1 se la frazione passata come parametro è maggiore dell'altra e 1 nel caso opposto.
7. Il metodo `stampa` che stampa a console l'oggetto frazione.
8. Il metodo `semplifica` che semplifica ai minimi termini la frazione. Il metodo cerca il massimo comun divisore tra `numeratore` e `denominatore` e divide entrambi per questo numero. Suggeriamo questo semplice algoritmo: si determina il numero minore tra numeratore e denominatore (sia `minimo`), quindi con un ciclo in cui si incrementa un indice intero da `1` a `minimo` si determina il massimo intero che divide esattamente sia numeratore che denominatore. Questo è il massimo comun divisore. L'algoritmo, così come descritto, funziona per numeri positivi. Una soluzione è quella di lavorare con i valori assoluti di numeratore e denominatore (`fabs` di `cmath`).

Scrivere un `main()` che legga il file di testo `numeri.txt` passando il nome del file dalla linea di comando.

Leggere le prime 10 righe del file, ciascuna contenente numeratore e denominatore di una frazione.

Costruire 10 oggetti frazione con i dati letti, stamparle con il metodo `stampa`, ridurle ai minimi termini con il metodo `semplifica` e stamparle nuovamente a schermo.

ATTENZIONE Si chiede di costruire 10 oggetti frazione differenti (quindi non di sovrascrivere nel ciclo di lettura del file uno stesso oggetto) da utilizzare nel seguito per testare gli altri metodi implementati nella classe. Suggerimento: è possibile definire un array di 10 oggetti di tipo frazione, oppure si può utilizzare la classe vector ...

Infine, testare i metodi implementati, eseguendo le seguenti operazioni sulle 10 frazioni precedentemente create, stampando ogni volta il risultato:

1. sommare tra loro le prime cinque frazioni;
2. dividere tale somma per la sesta frazione;
3. semplificare ai minimi termini il risultato dell'operazione precedente;
4. confrontare il risultato con la settima frazione;
5. moltiplicare il risultato per l'ottava frazione;
6. sottrare al risultato dell'operazione precedente la decima frazione.