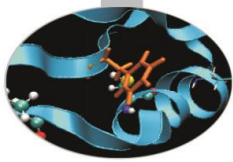
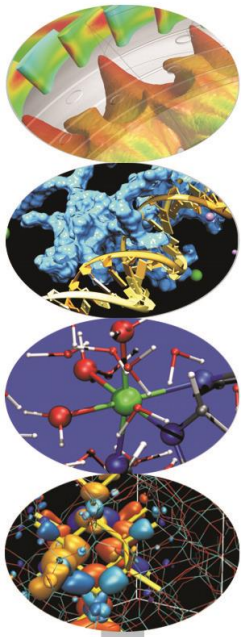
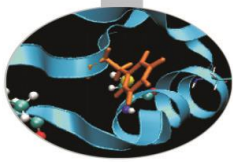


Esercitazioni



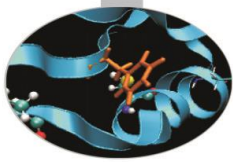


Esercizio 1

Riscrivere la classe *Complex* facendo l'overloading degli operatori: *>>*, *<<* (dichiarati come funzioni **friend**); *+*, *-*, *=*, *==*, *!=* (dichiarati come funzioni **membro**). Testare nel programma l'overloading di ciascuno di questi operatori.

Esercizio 2

Scrivere la classe *Array* ove un vettore, ricevuto in input dal programma, viene trasformato in un array di dimensioni specificate. La classe Array deve avere come dati membro private: il numero di righe e colonne della matrice, la dimensione della stessa ed un puntatore ad int, per es. *ptr_arr*.
Il costruttore deve ricevere solamente il numero di righe e colonne dell'array ed allocare abbastanza memoria perché ptr_arr possa contenere l'intero array. Fra i membri public predisporre il distruttore; il costruttore di copia; le funzioni **const** *getRows()* e *getColums()*; l'overloading degli operatori *==*, *!=*, *=*, e *( )*, quest'ultimo sia per la lettura che per la modifica di un elemento dell'array. Implementare anche l'overloading degli operatori *>>* e *<<* come funzioni **friend**.
Nel programma dichiarare almeno tre oggetti della classe Array per controllare il corretto funzionamento di tutti i metodi e delle funzioni friend.



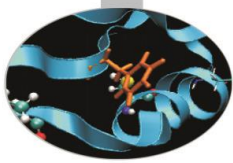
Esercizio 3

Creare una classe Atom che contiene i campi name, e le coordinate x,y,z. Definire gli operatori << e >> di inserimento ed estrapolazione.

Dichiarare un array di Atom nel main program, lungo N.

Inizializzare l'array leggendo i dati contenuti nel file xyz.dat, dove a ciascuna riga corrisponde un atomo.

Implementare la lettura da file utilizzando opportunamente la nuova classe e l'overloading degli operatori >> per la lettura e l'inizializzazione di Atom[i] e << per la stampa a video dell'array di Atom.



Esercizio 4

Definire la classe orologio che contiene l'ora del giorno (ore, minuti, secondi). Definire un costruttore e i seguenti metodi:

`void imposta(int ore, int minuti, int secondi);` per impostare l'ora

`void tick();` per incrementare l'ora di un secondo

`int secondi();` che restituisce il numero di secondi passati dalla mezzanotte

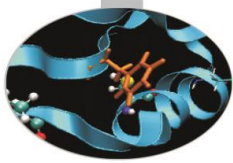
`getOra(int&, int&, int&);` che ritorna ore, minuti e secondi

Definire l'operatore << per stampare a video l'ora.

Definire una classe sveglia che eredita da orologio con il metodo

`void setSveglia(int ore,int minuti,int secondi);` per impostare la sveglia

`void tick();` che stampa DRIIIN se l'ora coincide con la sveglia (overriding).



Esercizio 5

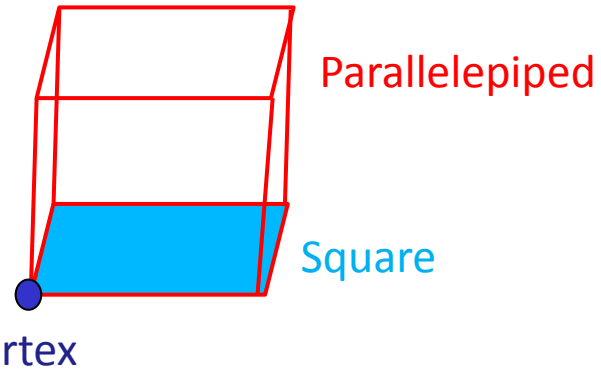
Basandosi sull'esempio cerchio->centro, definire la gerarchia di classi:
parallelepipedo->quadrato->vertice_in_basso_a_sinistra.

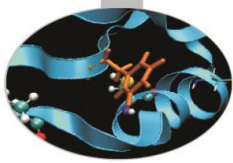
La classe vertex ha due attributi x,y (coordinate del vertice) e definisce l'operatore <<.
La classe quadrato eredita da vertex e possiede l'attributo lenght, per definire la lunghezza del lato, i metodi getArea e getPerimetro e ridefinisce l'operatore <<
La classe parallelepipedo eredita da quadrato e possiede l'attributo height, per definire l'altezza del parallelepipedo, i metodi getVolume e getArea e ridefinisce l'operatore <<

Esercizio 6

Riscrivere l'esercizio precedente utilizzando la composizione di classi

al posto dell'ereditarietà.





Esercizio 7:

Implementare la classe Persona con attributi sesso, età e nome. Con un metodo virtual chiSei();

Implementare la classe Impiegato che eredita da Persona con attributo stipendio. Con un metodo virtual chiSei();

Implementare la classe Dirigente che eredita da Impiegato con attributo bonus. Con uno metodo virtual chiSei(); Sperimentare il polimorfismo di classe.