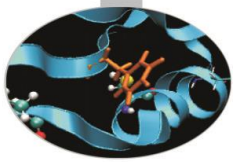
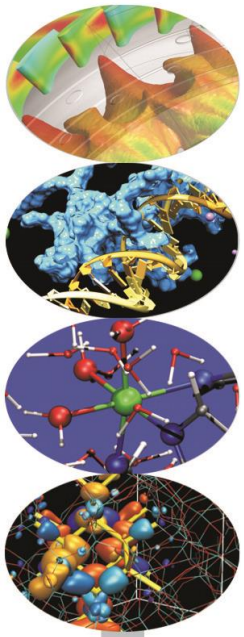
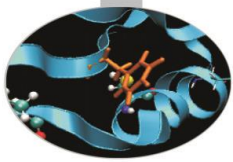


Esercitazione I





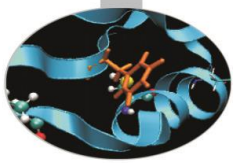
Esercizio 1

Costruire la classe Complex dei numeri complessi. I **dati membro private** sono rappresentati dalla **parte reale** e dalla **parte immaginaria** di un numero complesso.

Il **costruttore** venga scritto in modo tale da **inizializzare a zero sia la parte reale che la parte immaginaria**. Fra i **metodi public** inserire le funzioni **setReal()** e **setImg()**, per assegnare un valore non nullo sia alla parte reale che alla parte immaginaria, nonché le funzioni **addition()** e **subtraction()**, per eseguire l'addizione e la sottrazione di due numeri complessi, e **print_bra()** (da dichiarare come const) per stampare un numero complesso tra parentesi tonde.

Il programma deve richiedere come input la parte reale ed immaginaria di due numeri complessi e stampare su video, oltre ai due numeri, la loro somma e la loro differenza.

Esercizio 2



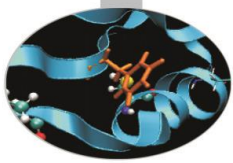
Costruire la classe **IntegerSet** (insieme di interi) il cui **unico membro private** è un **array di 10 elementi** (val_array[10]).

Un insieme di interi è rappresentabile attraverso un array che contenga soltanto i valori 1 e 0: le posizioni dell'array in cui è presente un 1 indicano gli interi dell'insieme.

Il **costruttore di IntegerSet** riceve come argomenti un vettore di interi e la sua dimensione; inizializza dapprima tutti gli elementi di val_array[10] a zero e, successivamente, pone uguale ad 1 gli elementi di val_array specificati dal vettore passato come parametro.

Continua…..

Esercizio 2

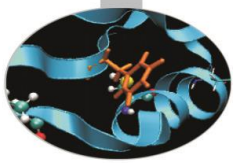


La classe deve contenere le funzioni membro public: **unionOfIntegerSets** ed **intersectionOfIntegerSets** che restituiscono (tramite il puntatore **this**) un terzo insieme di interi che rappresenti, rispettivamente l'unione e l'intersezione di due insiemi dati; **insertElement** e **deleteElement**, per aggiungere e cancellare un elemento da un insieme; **getVal_array**, per stampare su standard output il contenuto di val_array[10]. Utilizzare, poi,la funzione **friend isEqualTo**, per determinare se due insiemi sono uguali.

Tale funzione deve essere chiamata dal main(). Scrivere anche un costruttore di "default" che non riceva argomenti ed inizializzi semplicemente val_array[10] a zero (è utile per creare gli insiemi intersezione ed unione).

Dichiarare nel main() due oggetti della classe IntegerSet: set1_ e set_2, il cui contenuto viene poi determinato da due array di numeri interi compresi tra 0 e 9 scelti dal programmatore. Testare tutti i metodi della classe.

Esercizio 3

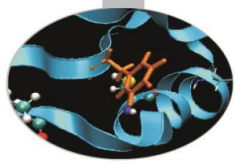


Scrivere la classe Date che comprenda i membri private *day*, *month* e *year* e soltanto i metodi public (dichiarabili come **const**) *getDay()*, *getMonth()*, *getYear()* oltre al costruttore.

Il costruttore deve verificare che i valori dei giorni, dei mesi e degli anni di un qualsiasi oggetto della classe Date rientrino nell'intervallo corretto, altrimenti li deve porre uguali a 1,1,2000 rispettivamente. Per gli anni si supponga di considerare solo l'intervallo 1900-2004, per esempio.

Considerare anche la possibilità che un anno sia bisestile (ovvero divisibile per quattro). Il programma deve contenere la dichiarazione di alcuni oggetti Date e stamparli su video.

Esercizio 4



Aggiungere alla classe Date una funzione incr che incrementi la data di un giorno ad ogni chiamata ed una funzione printDate() (da dichiarare **const**) per la stampa su video degli oggetti della classe Date. Supporre che gli anni debbano essere maggiori di 2000, per esempio.

Dichiarare nel main() un oggetto Date di partenza ed incrementarlo con incr per un numero di volte sufficiente a testare il corretto funzionamento del programma. Servono ancora le funzioni getDay(), getMonth() e getYear()?