Esercizi per il C++

1. Bubble sorter

Scrivi una classe BubbleSorter che ordina un array di interi.

BubbleSorter ha un **costruttore** che prende un array di interi tramite puntatore (int*) - e se serve la dimensione dell'array.

Ha un metodo **sort** che quando chiamato ordina l'array (**nota**: se non è stato già ordinato) senza però modificare l'array originale (deve quandi fare una copia).

Ha un metodo **getArrayOrdinato** che restituisce l'array ordinato.

Separa il .h e il .cpp di BubbleSorter e scrivi un file di prova con qualche array.

2. Scrivi una classe Complex Number – overload di operatori

Scrivi una classe complex che rappresenta numeri complessi. Complex ha tre costruttori (senza nulla, costruisce lo 0, con un argomento per numeri reali, e con due argomenti, parte reale e immaginaria). Prova con tre costruttori separati o con uno solo e argomenti di default.

Scrivi anche un metodo che restituisce la parte reale e immaginaria di un numero complesso.

Overload di operatori:

In C++ si posso ridefinire anche operatori non solo metodi. Ad esempio, posso ridefinire gli operatori +, - , ... in questo modo:

```
complex operator+(const complex & )const;
complex operator-(const complex & )const;
complex &operator=(const complex & );
complex operator*(const complex &)const;
bool operator==(const complex &)const;
bool operator!=(const complex &)const;
```

Prova ad implementare questi operatori.

Posso anche ridefinire gli operatori << e >> (che però appartengono alla classe ostream e istream) in questo modo:

```
//overloading di estrazione dello streaming
  friend ostream &operator<<( ostream & , const complex  &);
//overloading di immissione dello streaming
  friend istream &operator>>( istream &, complex & );
```

prova ad implementare << e >>.

Scrivi un main di prova in cui domandi (>>) 3 numeri complessi fai tutte le operazioni e stampi il risultato.

3. Funzioni Virtual, ereditarietà multipla

Dichiarate una classe Studente che eredita sia da Persona e da Discente. Ognuna delle due classi base ha qualche membro (campo e metodo) che viene ereditato (esempio **nome** per Persona, e **corso** a cui iscritto per Discente) e i cotruttori opportuni.

Studente ha due sottoclassi StudenteLS e StudenteIL con alcuni campi propri. Studente ha anche un costruttore che chiama quelli della superclasse.

Usate la classe std::string per le stringhe (con qualche operazione)

In Studente dichiarate un metodo virtual (getCorso) e uno non virtual (getNome) che poi ridefinite/override nelle sottoclassi. Entrambi restituiscono il dato con prefesso la classe che esegue il metodo. (Ad esempio "StudenteLS Informatica")

Nel main di prova provate a:

Creare 3 oggetti delle tre classi Studente.

Costruite un array di puntatori in cui mettete oggetti di tutte tre le classi

Chiamate il metodo virtuale e quello non virtuale per i tre oggetti

Fate lo stesso con un array di elementi (non puntatori). Cosa stampa?