Esercitazione 16 aprile

Classi e allocazione dinamica, overload di operatori

[Esercizio C061] [*prova completa del 7 giugno 2017, esercizio 5*]

Un'azienda ha otto dipartimenti. I dipendenti che lavorano in ciascun dipartimento possono appartenere ad una di cinque diverse classi stipendiali.

Allo scopo di gestire i dipendenti dell'azienda, si realizzi in linguaggio C++ la classe *Dipendenti* avente i seguenti attributi:

- **un array _*s* di cinque numeri reali** per rappresentare lo stipendio corrispondente a ciascuna delle cinque classi stipendiali e

- **una matrice di numeri interi _D di otto righe e cinque colonne** i cui elementi rappresentano, per ciascun dipartimento, il numero di dipendenti appartenenti a ciascuna classe stipendiale. Se, ad esempio, l'elemento nella terza riga e quarta colonna di _D valesse 10, significherebbe che nel terzo dipartimento lavorano 10 dipendenti appartenenti alla quarta classe stipendiale.

[Esercizio C061] [*prova completa del 7 giugno 2017, esercizio 5*]

Si implementino, inoltre, i seguenti metodi (1/2):

- Il costruttore di default che inizializzi a zero tutti gli elementi dell'array _*s* e della matrice _D.

- Il costruttore con parametri che riceva come parametri un array di cinque numeri reali e una matrice di numeri interi di otto righe e cinque colonne e li assegni rispettivamente all'array _*s* e alla matrice _D. Per semplicità si supponga che i valori passati al costruttore siano sempre validi.

- Il costruttore di copia.

- Il distruttore.

- I selettori per l'array _*s* e per la matrice _D:
  - Il selettore per _*s* restituirà come valore di ritorno un puntatore a costante all'array stesso.
  - Il selettore per _D riceverà in ingresso (ovvero come parametri) due numeri interi *h* e *k* e restituirà come valore di ritorno l'elemento della matrice _D di indici *h* e *k* (un numero intero). Nel caso in cui il valore di almeno uno dei due parametri *h* e *k* non fosse valido, il metodo restituirà -1.

[Esercizio C061] [*prova completa del 7 giugno 2017, esercizio 5*]

Si implementino, inoltre, i seguenti metodi (2/2):

- I modificatori per l'array _*s* e per la matrice _D. Per semplicità, si supponga che l'array passato come parametro al modificatore per _*s* e la matrice passata come parametro al modificatore per _D abbiano le dimensioni corrette e che i loro elementi assumano valori validi.

- Il metodo *totaleDipendenti* che calcoli e restituisca come valore di ritorno il numero totale dei dipendenti dell'azienda (un numero intero dato dalla somma degli elementi della matrice _D)

- Il metodo *estremiStipendio* che calcoli e restituisca come parametri di uscita (ovvero passati per riferimento) gli indici dei due dipartimenti (due numeri interi) per i quali l'azienda spende l'ammontare massimo e l'ammontare minimo per gli stipendi dei dipendenti. Il metodo non restituisce alcun valore di ritorno.

[Esercizio C061] [*prova completa del 7 giugno 2017, esercizio 5*]

Si scriva quindi la funzione *main* che operi nel modo seguente:

- Dichiari un array *stp* di cinque numeri reali e una matrice di numeri interi Dip di otto righe e cinque colonne.
- Apra in lettura il file *Dati.txt*. Tale file contiene nella prima riga cinque numeri reali che rappresentano gli stipendi associati a ciascuna classe stipendiale e nelle otto righe successive, per ciascuna riga, cinque numeri interi che rappresentano il numero di dipendenti inquadrati in ciascuna classe stipendiale per il dipartimento corrispondente alla riga in esame. Per semplicità si supponga che i valori contenuti nel file siano validi. Nel caso in cui il file *Dati.txt* non esista, *main* termina con valore di ritorno -1.
- Legga la prima riga del file *Dati.txt* assegnandone i valori agli elementi dell'array *stp* e le righe successive assegnandone i valori agli elementi della matrice Dip.
- Utilizzando il costruttore con parametri, allochi dinamicamente un oggetto *pdip* di tipo *Dipendenti*, passando l'array *stp* e la matrice Dip come parametri al costruttore.
- Invochi su tale oggetto i metodi *totaleDipendenti* ed *estremiStipendio* e stampi a video il risultato.
- Deallochi l'oggetto *pdip* e termini.

[Esercizio C061] [*prova completa del 7 giugno 2017, esercizio 5*]

Esempio di file Dati.txt

20000 24000 28000 32000 36000
6 3 2 1 0
2 3 4 1 1
3 5 8 2 1
3 4 7 3 2
8 2 0 1 1
5 4 3 2 1
6 7 2 1 3
5 4 2 2 4

[Esercizio C062] Overload di operatori

Implementare la classe *punto3d* con i seguenti metodi:

- Il costruttore di default che inizializza a zero tutti i attributi,
- Un costruttore con parametri che riceve in ingresso (ovvero come parametri) tre numeri reali che rappresentano le coordinate *x, y e z* di un punto 3D e assegna i valori dei parametri attuali ai corrispondenti attributi della classe
- Il costruttore di copia
- Il distruttore
- I selettori (un metodo per ciascun attributo)
- I modificatori (un metodo per ciascun attributo)

[Esercizio C062] Overload di operatori

Implementate i seguenti operatori:

- L'operatore unario di cambio di segno, per calcolare l'opposto di un punto 3D.
- Gli operatori unari di incremento e decremento unitario prefisso e postfisso.
- Gli operatori aritmetici di somma, sottrazione e moltiplicazione per scalare.
- L'operatore di assegnamento.
- L'operatore relazionale di uguaglianza ==, per il quale due punti 3D sono uguali se hanno le stesse coordinate.
- Gli operatori di inserimento ed estrazione.
- Un funzione esterna amica per il calcolo della distanza tra due punti.

Si scriva quindi un programma C++ per verificare il corretto funzionamento della classe sviluppata.
Il programma dichiarerà due oggetti di classe *punto3D*, chiederà all'utente di immettere da tastiera i valori delle coordinate dei due punti utilizzando l'operatore di inserimento, stamperà a video le coordinate immesse utilizzando l'operatore di estrazione e calcolerà e stamperà a video la somma, la differenza, e la distanza tra i due punti.

[Esercizio C063] Utilizzo di classe Punto3D

Assumiamo che il Punto3D corrisponde alla posizione della mano misurata con il sistema di cattura di movimento.
Il sistema è in grado di rilevare 15 volte al secondo la posizione della mano. Implementare un programma in C++ che:

- calcola la velocita della mano sulle finestre di 15 misurazioni (1 secondo).
- legga da un file di testo un insieme di punti 3D e calcoli la velocità media su finestre di 15 punti utilizzando la tecnica di finestra scorrevole.

Il programma deve:

- aprire il file di input contenente i punti 3D.
- allocare dinamicamente di un array di punti 3D.
- leggere dei punti dal file e inserire nell'array dinamico.
- calcolare la velocità media su finestre di 15 punti utilizzando la tecnica di finestra scorrevole.
- stampare la velocità media per ogni finestra di 15 punti sulla console.

[Esercizio C063] Utilizzo di classe Punto3D

Esempio del file punti3D.txt:

```
0 0 0
0.000109739 0.000109739 0.000109739
0.000548697 0.000548697 0.000548697
0.00153635 0.00153635 0.00153635
0.00329218 0.00329218 0.00329218
0.00603567 0.00603567 0.00603567
0.00998628 0.00998628 0.00998628
0.0153635 0.0153635 0.0153635
0.0223868 0.0223868 0.0223868
0.0312757 0.0312757 0.0312757
0.0422497 0.0422497 0.0422497
0.0555281 0.0555281 0.0555281
```

[Esercizio C064] Overload di operatori

Progettare e sviluppare in linguaggio C++ la classe *poligono2d* per rappresentare poligoni i cui vertici sono punti bidimensionali. Il poligono è rappresentato come un array di punti bidimensionali, ovvero un array di oggetti della classe *punto2d (per il codice della classe punto2d vedi il codice Esercizio14_1 su github)*.

La classe avrà pertanto i seguenti attributi:

- il puntatore *_p* ad un array di oggetti di classe *punto2d*
- il numero *_n* di punti contenuti nell'array (un numero intero).

Si implementino, inoltre, i seguenti metodi (1/4):

- Il costruttore di default che inizializzi *_p* a NULL e *_n* a zero.

- Un costruttore con parametri che riceva in ingresso (ovvero come parametri) un numero intero *dim*. Il costruttore effettuerà l'allocazione dinamica di un array di *dim* oggetti di classe *punto2d*, ne assegnerà il puntatore a *_p* e assegnerà *dim* a *_n*. Nel caso in cui l'allocazione dinamica della memoria non vada a buon fine o il valore di *dim* non sia valido, il costruttore inizializzerà *_p* a NULL e *_n* a zero.

[Esercizio C064] Overload di operatori

Si implementino, inoltre, i seguenti metodi (2/4):

- Un costruttore con parametri che riceva in ingresso (ovvero come parametri) un array *pti* di oggetti di classe *punto2d* e la sua dimensione *dim* (un numero intero). Il costruttore opererà come segue:
    - nel caso in cui *dim* assuma un valore valido, effettuerà l'allocazione dinamica di un array di *dim* oggetti di classe *punto2d* e ne assegnerà il puntatore a *_p*;
    - nel caso in cui l'allocazione dinamica della memoria vada a buon fine, assegnerà a ciascun elemento dell'array puntato da *_p*, il valore del corrispondente elemento dell'array *pti* e assegnerà *dim* a *_n*;
    - nel caso in cui l'allocazione dinamica della memoria non vada a buon fine o il valore di *dim* non sia valido, il costruttore inizializzerà *_p* a NULL e *_n* a zero.

- Il costruttore di copia. Nel caso in cui l'allocazione dinamica della memoria non vada a buon fine, il costruttore inizializzerà *_p* a NULL e *_n* a zero.

- Il distruttore.

- I selettori (un metodo per ciascun attributo).

[Esercizio C064] Overload di operatori

- Il modificatore per l'array puntato da *_p* che riceva in ingresso (ovvero come parametri) un array *pti* contenente i nuovi valori e la sua dimensione *dim* e operi come segue:
  - nel caso in cui il valore di *dim* sia valido, dichiari un puntatore *p* a un array di oggetti di classe *punto2D*, effettui l'allocazione dinamica di un array di *dim* oggetti di classe *punto2D* e ne assegni il puntatore a *p*,
  - nel caso in cui l'allocazione dinamica della memoria vada a buon fine, assegni a ciascun elemento dell'array puntato da *p*, il valore del corrispondente elemento dell'array *a*, deallochi (se esiste) l'array puntato da *_p*, assegni il valore di *p* al puntatore *_p* e assegni *dim* a *_n*;
  - nel caso in cui l'allocazione dinamica della memoria non vada a buon fine o il valore di *dim* non sia valido, il modificatore non opera alcuna modifica ai dati.

- L'operatore di moltiplicazione per scalare,
- L'operatore di assegnamento. Nel caso in cui l'allocazione dinamica della memoria non vada a buon fine, l'operatore assegnerà NULL a *_p* e zero a *_n*,
- L'operatore di selezione con indice,
- L'operatore relazionale di uguaglianza, per il quale due poligoni sono uguali se sono uguali le coordinate di tutti i punti che ne costituiscono i vertici,

[Esercizio C064] Overload di operatori

Si implementino, inoltre, i seguenti metodi (4/4):

- Gli operatori di inserimento ed estrazione.
- Un metodo per calcolare il perimetro del poligono come somma delle distanze tra coppie di vertici consecutivi.

Si scriva infine un programma C++ per verificare il corretto funzionamento della classe sviluppata.

Il programma:

- chiederà all'utente di inserire da tastiera il numero di vertici desiderato per il poligono,
- dichiarerà un oggetto di classe *poligono2d* passando al costruttore il numero di vertici desiderato,
- chiederà quindi all'utente di inserire da tastiera le coordinate dei vertici del poligono utilizzando l'operatore di estrazione,
- calcolerà infine il perimetro del poligono e ne stamperà a video il valore.