Esercizi di programmazione in linguaggio C++ – Classi

Nota: memorizzare i programmi in cartelle separate. Ogni cartella deve contenere, oltre al codice sorgente e al file eseguibile, gli eventuali file di input indicati nel testo e file di output prodotti dal programma. I file di input devono essere generati con un apposito editor prima della scrittura del codice.

Definizione di classi

1. **persona** – Creare una classe *Persona* per memorizzare cognome, nome ed età di una persona. La classe deve disporre di un opportuno costruttore, di metodi che consentano l'accesso in lettura a tutti gli attributi, di un metodo per l'eventuale modifica dell'età e del metodo *stampa_anagrafica* per la stampa a video di tutte le informazioni sulla persona. Successivamente, scrivere un programma che legga da tastiera i dati di due persone, li memorizzi in oggetti distinti e indichi se le due persone sono coetanee. In caso contrario, il programma deve stampare i dati anagrafici della persona più giovane. Rispettare i principi dell'*information hiding* (attributi privati, metodi pubblici).

2. **veicolo** – Scrivere la classe *Veicolo* avente i seguenti attributi: marca, modello, velocità in km/h, stato del motore (acceso/spento). Tutti gli attributi devono essere accessibili in lettura. La classe fornisce inoltre i seguenti metodi:

   - *accelera(d)*: aumenta la velocità del veicolo acceso di un valore *d* espresso in km/h (il valore può essere anche negativo). Si suppone che il veicolo non possa superare la velocità di 200 km/h.
   - frena(): arresta il veicolo in movimento.
   - *accendi()* e *spegni()*: metodi impiegati per accendere e spegnere il veicolo (ogni metodo controlla lo stato del motore e la velocità prima di eseguire la propria azione).

   Scrivere un programma che, dopo aver inizializzato un oggetto di classe *Veicolo*, permetta attraverso un apposito menù di condurre l'autoveicolo (avvio e arresto del motore, azione su freno o acceleratore). Nel menù devono inoltre essere presenti comandi per mostrare i singoli parametri del veicolo (la classe fornisce il metodo di accesso, il *main* si occupa della stampa a video del valore).

3. **tabellone** – Realizzare una classe per la gestione del tabellone e del sacchetto di numeri della tombola. La classe deve possedere opportuni attributi per rappresentare il contenuto del sacchetto e conservare lo stato (estratto/non estratto) di ciascun numero. La classe deve inoltre fornire i seguenti metodi:

   - *nuovo()*: inizializza il tabellone e colloca nel sacchetto tutti i numeri;
   - *estrai()*: estrae un numero a caso dal sacchetto e ne restituisce il valore (può essere utile ricorrere all'algoritmo di generazione di numeri casuali distinti visto in passato);
   - *completato()*: restituisce *true* se tutti i numeri della tombola sono stati estratti dal sacchetto.
   - *stampa_estrazioni()*: stampa a video la sequenza di numeri estratti dal sacchetto (in ordine di estrazione);
   - *stampa_tab()*: stampa a video il tabellone, evidenziando in particolare i numeri estratti. Un possibile esempio di stampa è il seguente:

     ```
     ** **  3 ** ** ** **  8 ** **
     ** ** ** 14 ** 16 ** ** ** 20
     ```

   Realizzare un programma che utilizzi la classe per simulare l'estrazione di tutti i numeri dal sacchetto, mostrando alla fine di ogni estrazione il tabellone aggiornato. Dopo aver estratto l'ultimo numero, il programma deve elencare tutti i numeri in ordine di estrazione.

4. **tessera** – Realizzare una classe per la gestione della tessera di una tombola. Una tessera è caratterizzata dal nome del giocatore e da una riga formata da 5 numeri interi, distinti e casuali, compresi tra 1 e 90. Ogni numero è idealmente inserito in una finestrella che può assumere lo stato aperta (numero non ancora estratto) oppure chiusa (numero già estratto). La classe deve disporre di appositi attributi per rappresentare lo stato delle finestrelle e deve fornire i seguenti metodi:

   - *nuova()*: Inizializza le finestrelle della tessera ponendole tutte nello stato aperta;
   - *controlla(n)*: verifica la presenza del numero n, ricevuto come parametro di input, all'interno della tessera. Se il numero è presente, il metodo chiude la finestrella corrispondente. Il metodo restituisce true se il numero è presente nella tessera, false in caso contrario;
   - *completa()*: restituisce true se e solo se tutte le finestrelle della tessera sono chiuse, false in casso contrario;
   - *stampa()*: stampa il nome del giocatore e i numeri della tessera, secondo la seguente convenzione: se una finestrella associata è aperta, il metodo stampa il numero corrispondente; se la finestrella è chiusa, stampa la sequenza ##. Esempio:

     ```
     Mario:    ##  34  ##  ##  67
     ```

   La classe deve avere inoltre un costruttore che inizializzi una tessera dato il nome di un giocatore.

   Successivamente, scrivere un programma che simuli l'uso di una tessera durante il gioco della tombola. In particolare il programma, dopo aver chiesto l'inserimento di un nome da tastiera, deve creare un'istanza della classe. In seguito, il programma chiede l'inserimento di un numero, indichi se il numero è presente nella tessera e riporti su monitor lo stato della tessera del giocatore. Queste operazioni si ripetono fino alla conclusione del gioco (cioè fino a quando tutte le finestrelle risultano chiuse).

5. **punto** –
6. **segmento** –

Ereditarietà

7. **articoli1** – Gli articoli venduti da un negozio sono caratterizzati da un codice a barre, una descrizione e un prezzo unitario. Scrivere una classe **Articolo** adatta a memorizzare tali informazioni, includendo opportuni costruttori e metodi *accessor*. Definire inoltre un metodo *sconta()* che modifichi il prezzo dell'articolo diminuendolo del 5%. Scrivere un main che chieda l'inserimento di 3 articoli da tastiera e stampi il costo totale.
8. **articoli2** – Il negozio desidera suddividere gli articoli in *alimentari* e *non alimentari*. Agli articoli alimentari è associato un anno di scadenza, a quelli alimentari il materiale principale di cui sono fatti. Realizzare le classi derivate **ArticoloAlimentare** e **ArticoloNonAlimentare**, ciascuna con l'opportuno costruttore e un metodo *accessor* per il nuovo attributo, ridefinendo il metodo sconta() in base alle seguenti regole:

   - per gli articoli alimentari, lo sconto è del 20% se l'anno di scadenza coincide con quello attuale;
   - per gli articoli non alimentari, lo sconto è del 10% se il materiale è riciclabile (vetro, carta o plastica).

   Realizzare un programma che chieda l'inserimento da tastiera di 3 articoli alimentari e 2 articoli non alimentari, scontando i prezzi se il cliente dispone della tessera fedeltà. Terminato l'inserimento, il programma deve stampare lo scontrino contenente il nome degli articoli, il loro prezzo unitario (eventualmente scontato) e l'importo totale da pagare.
9. **squadre** – Creare una classe *Squadra* che rappresenti una squadra di calcio e abbia come attributi il nome della squadra, il numero di partite vinte, il numero di partite perse e il numero di partite pareggiate. La classe deve

disporre di opportuni metodi per impostare o visualizzare gli attributi, un metodo *punti()* che restituisce quanti punti ha in campionato (ogni partita vinta vale 3 punti, ogni partita pareggiata 1, quelle perse 0) e un metodo *inizioanno()* che azzera il numero di partite vinte, pareggiate e perse. Creare un *main* per provare la classe creando due istanze (Juventus e Milan), facendo inserire all'utente per entrambe le squadre il numero di partite vinte, perse e pareggiate e indicando la squadra che ha ottenuto più punti in campionato.

10. **punto** –

11. **puntospazio** –

12. **recinzioni** – Una ditta specializzata nel commercio di prodotti per l'edilizia vende due modelli di recinzioni per abitazioni: il modello BASIC al costo di 45 Euro al metro e il modello PLUS al costo di 60 Euro al metro.

    La ditta vuole utilizzare un software che consenta di inviare automaticamente a ogni cliente una e-mail con il preventivo per l'acquisto della recinzione. Per questo motivo, realizza un file di testo *clienti.txt* contenente le seguenti informazioni: codice identificativo (numerico); indirizzo e-mail del cliente, coordinate GPS dell'abitazione (due numeri reali) e modello desiderato. Un esempio di record è il seguente:

    ```
    176551  mario.rossi@gmail.com  41.599  12.873  PLUS
    ```

    Per rilevare le misure, la ditta utilizza un drone che, giunto all'abitazione del cliente grazie alle coordinate GPS, inizia a riprenderne i confini, li trasforma in un poligono mediante un apposito software di riconoscimento e ne rileva le coordinate X,Y (in metri) dei vertici. I vertici sono inizialmente memorizzati nel disco del drone, in un file di testo il cui nome coincide con il codice identificativo del cliente (es. 176551.txt) e successivamente trasferiti nel server centrale.

    Dopo aver inserito nel file *clienti.txt* le informazioni di almeno tre clienti e generato i corrispondenti file dei vertici, scrivere un'applicazione che, utilizzando la classe *Punto* realizzata negli esercizi precedenti, consenta di calcolare automaticamente il perimetro di ciascuna recinzione e determinare il costo totale considerando il modello acquistato. I risultati dell'elaborazione devono essere memorizzati nel file di testo *preventivi.txt* secondo il formato: codice identificativo; e-mail; modello; lunghezza totale in metri (arrotondata a tre cifre decimali); costo in Euro. Un esempio di record di output è il seguente:

    ```
    176551  mario.rossi@gmail.com  PLUS  15.678  940.68
    ```

Esercizio n. 6 (segmento)

```
//            File: segmento.cpp
// Ultima modifica: 06-04-2017

#include <iostream>
#include <string>
#include <cmath>
#include <iomanip>      // setprecision

using namespace std;

class Punto {
    private:        // Modificatore di accesso
        string nome;
        double x;
        double y;

    public:
        Punto(string n, double coord_x, double coord_y);
        string getNome();
        double getX();
        double getY();
        void trasla(double dx, double dy);
        double distanza();
        void stampa();
};

class Segmento {
    private:
        Punto p1, p2;

    public:
        Segmento(string n1, double coord_x1, double coord_y1, string n2,
                double coord_x2, double coord_y2);
        string getNome();
        void trasla(double dx, double dy);
        double lunghezza();
        void stampa();
};

Punto::Punto(string n, double coord_x, double coord_y) {
    this->nome = n;
    this->x = coord_x;
    this->y = coord_y;
}

string Punto::getNome() {
    return this->nome;
}

double Punto::getX() {
    return this->x;
```

```
}

double Punto::getY() {
    return this->x;
}

void Punto::trasla(double dx, double dy) {
    this->x += dx;
    this->y += dy;
}

double Punto::distanza() {
    return sqrt(this->x * this->x + this->y * this->y);
}

void Punto::stampa() {
    cout << this->nome << "(" << this->x << "; " << this->y << ")";
}

Segmento::Segmento(string n1, double coord_x1, double coord_y1, string n2, double coord_x2, double coord_y2) : p1(n1, coord_x1, coord_y1), p2(n2, coord_x2, coord_y2) {}  // Lista di inizializzazione

string Segmento::getNome() {
    return p1.getNome() + p2.getNome();
}

void Segmento::trasla(double dx, double dy) {
    p1.trasla(dx, dy);
    p2.trasla(dx, dy);
}

double Segmento::lunghezza() {
    double diff_x = p1.getX() - p2.getX();
    double diff_y = p1.getY() - p2.getY();

    return sqrt(pow(diff_x, 2) + pow(diff_y, 2));
}

void Segmento::stampa() {
    cout << "Segmento " << p1.getNome() << p2.getNome() << " da " ;
    p1.stampa();
    cout << " a ";
    p2.stampa();
    cout << " di lunghezza " << this->lunghezza();
}

int main() {
    cout << fixed << setprecision(2);   // Formato fisso, due cifre dopo la virgola

    Segmento s("A", 3.0, 5.0, "B", 7.0, 8.0);

    cout << "Prima della traslazione: ";
    s.stampa();
```

```
    s.trasla(4.0, 9.0);

    cout << endl << "Dopo la traslazione: ";
    s.stampa();

    return 0;
}
```

Esercizio n. 9 (squadre)

```
//      File: squadre.cpp
//
//      Gestione dei punti totalizzati da una squadra di calcio

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

const int PUNTI_PER_VITTORIA = 3;
const int PUNTI_PER_PAREGGIO = 1;
const int PUNTI_PER_SCONFITTA = 0;

class Squadra {
private:
    string nome;
    int vinte;
    int pareggiate;
    int perse;

public:
    Squadra();
    Squadra(string n);

    // Accessor e mutator di tipo inline
    string get_nome() { return nome; }
    void set_nome(string n) { nome = n; }
    int get_vinte() { return vinte;    }
    void set_vinte(int v) { vinte = v; }
    int get_pareggiate() { return pareggiate;    }
    void set_pareggiate(int p) { pareggiate = p; }
    int get_perse() { return perse;    }
    void set_perse(int p) { perse = p; }

    int punti();
    void inizioanno();
};

Squadra::Squadra() :
    nome(""), vinte(0), pareggiate(0), perse(0) {}

Squadra::Squadra(string n) :
    nome(n), vinte(0), pareggiate(0), perse(0) {}

int Squadra::punti() {
    return  vinte * PUNTI_PER_VITTORIA +
            pareggiate * PUNTI_PER_PAREGGIO +
            perse * PUNTI_PER_SCONFITTA;
}
```

```
void Squadra::inizioanno() {
    vinte = 0;
    pareggiate = 0;
    perse = 0;
}

int main() {
    Squadra s1("Juventus");
    Squadra s2("Milan");
    char scelta;

    do {
        int v, p, s;

        cout << "Numero di vittore, pareggi e sconfitte per la squadra '" <<
            s1.get_nome() << "': ";
        cin >> v >> p >> s;

        s1.set_vinte(v);
        s1.set_pareggiate(p);
        s1.set_perse(s);

        cout << "Numero di vittore, pareggi e sconfitte per la squadra " << s2.get_nome() << ": ";
        cin >> v >> p >> s;

        s2.set_vinte(v);
        s2.set_pareggiate(p);
        s2.set_perse(s);

        cout << "Risultato finale" << endl;
        cout << "----------------" << endl;
        cout << setw(15)  << s1.get_nome() << "\t" << s1.punti() << " punti" << endl;
        cout << setw(15) << s2.get_nome() << "\t" << s2.punti() << " punti" << endl << endl;

        if ( s1.punti() != s2.punti() )
            cout << "Campionato vinto dalla squadra " <<
                ( s1.punti() > s2.punti() ? s1.get_nome() : s2.get_nome() ) << endl;
        else
            cout << "Situazione di pari merito, e' necessario procedere allo spareggio" << endl;

        cout << endl << "Continuare con un nuovo campionato [S/N] ? ";
        cin >> scelta;

        s1.inizioanno();
        s2.inizioanno();

    }
    while (scelta == 'S' || scelta == 's');

    return 0;
}
```

File di testo per l'esercizio n. 12 (recizioni)

clienti.txt

```
176551  mario.rossi@gmail.com  41.599  12.873  PLUS
176552  paola.bianchi@azienda.it  42.012  12.754  BASIC
176553  carlo.neri@abc.it  41.158  13.124  PLUS
```

176551.txt

```
2.654  18.556
8.444  23.927
9.576  26.239
9.587  31.445
```

176552.txt

```
-6.884  -4.123
9.556  8.771
12.887  19.444
16.443  20.370
12.883  27.772
10.338  32.561
```

176553.txt

```
71.667  34.872
56.237  26.446
25.328  19.667
18.558  11.234
7.556   -6.982
-3.212  -12.967
```