Prof. Marco Camurri

Capitolo 15 - Classi

1. La classe Rettangolo

Definire la classe Rettangolo, caratterizzata dagli attributi altezza e larghezza. Nota: altezza e larghezza sono due numeri reali, e la classe Rettangolo dovrà garantire che essi non assumano mai valori negativi. La classe deve contenere:

a. un costruttore senza parametri che inizializza le dimensioni a zero

b. un costruttore con due parametri che consente di specificare altezza e larghezza iniziali (se il valore fornito è negativo l'attributo è impostato a zero)

c. il distruttore, che stampa il messaggio "oggetto distrutto"

d. metodi per ottenere e modificare il valore degli attributi

e. metodi per ottenere l'area e il perimetro del rettangolo

f. il metodo **void stampa()** che scrive le dimensioni del rettangolo

g. il metodo **void acquisisci()** che richiede all'utente i valori di altezza e larghezza (con controllo dell'input) e imposta le proprietà

Utilizzando tutti i metodi precedenti, scrivere un programma che acquisisce le dimensioni di due rettangoli, e comunica quale rettangolo ha l'area maggiore.

---

2. La classe Punto

Definire la classe Punto, caratterizzata dalle coordinate x e y di tipo float, e dotata di:

a. un costruttore vuoto

b. un costruttore con due parametri che consente di inizializzare x e y

c. un distruttore che stampa il messaggio "Punto (x,y) distrutto"

d. i metodi di accesso alle proprietà

e. un metodo per acquisire le coordinate di un punto dall'utente

f. un metodo che stampa le coordinate del punto

g. un metodo che restituisce la distanza fra il punto corrente ed un altro punto

h. un metodo che restituisce *true* se il punto corrente coincide con un altro punto

Infine, scrivere un main che richiede all'utente le coordinate di due punti non coincidenti (se il secondo punto coincide con il precedente il programma deve chiedere di nuovo le coordinate del secondo punto). Dopo aver acquisito le coordinate dei due punti, il programma deve mostrare la loro distanza.
Ricorda che la distanza tra due punti (X1,Y1) e (X2,Y2) è data dalla formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

---

3. La classe Frazione

Definire la classe Frazione, caratterizzata dai numeri interi numeratore (n) e denominatore (d). Il denominatore deve essere sempre diverso da zero. Inoltre la frazione deve risultare in ogni istante ridotta ai minimi termini (semplificata). Aggiungere alla classe:

a. un costruttore senza parametri, che inizializza la frazione al valore 1/1

b. un costruttore con due parametri che consenta di impostare numeratore e denominatore (vedi nota 1)

c. i metodi get e set per numeratore e denominatore (vedi nota 1)

d. il metodo **void input()** che richiede all'utente numeratore e denominatore (con controlli)

e. un metodo per stampare la frazione come spiegato negli esempi seguenti. Esempio 1: se num=1 e den=2 il metodo stamperà "1/2". Esempio 2: se num=5 e den=1 la frazione deve essere stampata come "5" e non "5/1". Esempio 3: se num=1 e den=-2 la frazione deve essere stampata come "-1/2" e non come "1/-2".

f. il metodo privato **void semplifica()** che riduce ai minimi termini la frazione corrente (vedi nota 2).

g. il metodo **void aggiungi(Frazione& f2)** che modifica il valore della frazione corrente, sommandovi la frazione f2

h. il metodo **void sottrai(Frazione& f2)** che modifica il valore della frazione corrente, sottraendovi la frazione f2

i. il metodo **void moltiplicaPer(Frazione& f2)** che modifica il valore della frazione corrente, moltiplicandola per f2

j. il metodo **void dividiPer(Frazione& f2)** che modifica il valore della frazione corrente, dividendola per f2

k. il metodo **int confrontaCon(Frazione& f2)** che restituisce un valore negativo se la frazione corrente è minore di **f2**, 0 se le due frazioni sono uguali, e un valore positivo se la frazione corrente è maggiore di **f2**

Scrivere un programma che consenta di testare tutti i metodi.

Nota 1: se si tenta di impostare un denominatore nullo, la classe stampa il messaggio "Errore divisione per zero" e termina il programma invocando la funzione exit(-1).
Nota 2: per ridurre ai minimi termini una frazione si dividono numeratore e denominatore per il Massimo Comun Divisore (MCD) calcolato con la seguente funzione:

```
int MCD(int a, int b) {
  if (a<0) a=-a;
  if (b<0) b=-b;
  if (b == 0) return a;
  int r = a % b;
  while(r != 0) {
    a = b;
    b = r;
    r = a % b;
   }
   return b;
}
```

3b. Calcolatrice con frazioni

Utilizzando la classe definita nell'esercizio 3, scrivere un programma che implementi una calcolatrice per il calcolo con le frazioni, come mostrato nell'esempio.

```
Inserisci frazione: 1/2
Inserisci operazione: +
Inseisci frazione: 1/3
Risultato: 5/6
Inserisci operazione: *
Inserisci frazione: 6/5
Risultato: 1
Inserisci operazione: q
Fine.
```

Come mostrato nell'esempio, l'utente immette una frazione, quindi sceglie l'operazione (+,-,*,/) e infine immette la seconda frazione da utilizzare nel calcolo. Il programma mostra il risultato, e richiede un'ulteriore operazioni da eseguire su di esso. L'utente esce dal programma premendo il tasto q.

---

4. La classe Data

Definire la classe Data, caratterizzata da giorno, mese e anno, e dotata di:

a. un costruttore senza parametri, che inizializza la data al 1 gennaio 1970 (giorno=1, mese=1, anno=1970)

b. un costrutture a tre parametri che permetta di impostare giorno, mese e anno. Se i valori passati non rappresentano una data valida, il costrutture deve stampare un messaggio di errore e terminare il programma invocando la funzione exit

c. il metodo **void acquisisci()** che consente di impostare giorno mese e anno acquisendoli dall'utente nel formato gg-mm-aa (vedi esempio). Il metodo deve impedire l'inserimento di date non valide. Per stabilire se un anno è bisestile vedi la nota.

d. i metodi get e set per anno, mese e giorno. Se si tenta di impostare una data non valida, la data corrente deve rimanere immutata

e. il metodo **void stampa()** che stampa la data nel formato gg-mmm-aaaa (esempio: "01-gen-2010")

f. il metodo **void aggiungiGiorni(int N)** che consente di aggiungere **N** giorni alla data. Suggerimento: scrivere prima un metodo privato che aggiunge un solo giorno alla data corrente, dopodichè scrivere il metodo generale richiamando N volte il metodo privato.

g. il metodo **int confrontaCon(Data& data2)** che restituisce -1 se la data è precedente a **data2**, 0 se le due date sono uguali, e +1 se la data è maggiore di **data2**

h. il metodo **int distanzaDa(Data& data2)** che restituisce il numero di giorni di distanza tra l'oggetto corrente e la data2. Suggerimento: per calcolare la distanza invocare ripetutamente il metodo aggiungiGiorni(1) sulla data minore, fino ad arrivare alla data maggiore. Il numero di invocazioni effettuate è uguale alla distanza in giorni tra le due date.

Utilizzando la classe Data, scrivere un programma che:

- richiede all'utente una data e numero intero N, e stampa la data che si trova N giorni dopo

- richiede un'ulteriore data, comunica se è maggiore o minore della precedente, e calcola la distanza in giorni fra le due date

Esempio:

```
Inserisci data: 30/12/2018
La data inserita e' 30-dic-2018
Inserisci un numero intero: 4
La data dopo 4 giorni e' 3-gen-2018
Inserisci data: 10/01/2018
La differenza fra 30-dic-2018 e 10-gen-2018 e' di 11 giorni
```

Nota: si ricorda che un anno è bisestile se e solo se è vera una di queste due condizioni:
a) l'anno è divisibile per 4 e non per 100
b) l'anno è divisibile per 400

---

5. La classe Orario

Si necessita di una classe adatta a rappresentare un orario. Un orario è caratterizzato dai suoi valori di ore, minuti e secondi. Le ore variano nell'intervallo 0..23, mentre minuti e secondi variano nell'intervallo 0..59. Ad esempio, l'ultimo orario valido della giornata è quindi 23:59:59. L'orario successivo è 00:00:00. Definire la classe fornendo:

a. un costruttore senza parametri, che inizializza ore, minuti e secondi al valore 0.

b. un costruttore che inizializza ore, minuti e secondi ai valori forniti dai parametri.

c. il metodo **void acquisisci()** che consente di acquisire dall'utente un orario nel formato hh:mm:ss. Il metodo deve garantire che i valori acquisiti formino un'orario valido.

d. il metodo **void stampa()** che stampa l'orario nel format hh:mm:ss

e. il metodo **void avanza(int n)** che fa avanzare l'orario di n secondi. Il valore di n può essere maggiore di 59.

f. metodi per impostare e ottenere i valori attuali di ore, minuti e secondi. Se si tenta di impostare un valore non valido, l'orario attuale rimane immutato

g. il metodo **int confronta(Orario& o2)** che consenta di confrontare l'orario corrente con l'orario o2, restituendo 0 se sono uguali, un valore positivo se l'orario attuale è maggiore, e un valore negativo se l'orario attuale è minore.

h.

i. il metodo **int differenza(Orario& o2)** che restituisce la differenza in secondi tra i due orari

Scrivi un programma che richiede all'utente un orario e un numero di secondi da sommarvi, e mostra il nuovo orario. Esempio:

```
Inserisci un orario: 9:59:59
Hai inserito: 09h:59m:59s
Quanti secondi vuoi sommare? 2
Nuovo orario: 10h:00m:01s
La differenza fra il secondo orario ed il primo è di 2 secondi
```

---

6. La classe Sveglia

Utilizzando la classe Orario dell'esercizio precedente, scrivi una classe che rappresenta una sveglia.
La sveglia consente in ogni momento di leggere l'orario attuale. Inoltre essa consente di impostare fino a 3 allarmi (allarme 0, allarme 1 e allarme 2), ognuno dei quali può essere in qualsiasi momento attivato o disattivato. Appena l'orario attuale coincide con uno degli orari di allarme, la sveglia inizia a suonare e continua per 1 minuto. Definisci la classe Sveglia dotata dei seguenti metodi:

a. **void impostaAttuale(Orario o)** imposta l'ora attuale

b. **void avanza()** fa avanzare di un secondo l'orario attuale della sveglia

c. **void impostaAllarme(int n, Orario o)** imposta l'orario in cui suonerà l'allarme numero n

d. **void impostaStatoAllarme(int n, bool stato)** attiva o disattiva l'allarme numero n

e. **bool staSuonando()** restituisce *true*se, a causa dell'orario attuale e degli allarmi impostati, la sveglia sta attualmente suonando. Ricorda che la sveglia suona per 1 minuto dall'inizio di ogni allarme attivo.

f. **void stampa()** stampa lo stato attuale della sveglia. Ad esempio, se attualmente la sveglia segna le 8:03 e vi è un solo allarme attivo per le ore 8.00, allora il messaggio da mostrare è:

```
ORA MOSTRATA: 08h:03m:00s
ALLARMI: [08h:00m:00s,ON] [00h:00m:00s,OFF] [00h:00m:00s,OFF]
STA SUONANDO: SI
```

Utilizzando le classi Orario e Sveglia, scrivi un programma che consenta all'utente di impostare l'ora attuale segnata dalla sveglia, e successivamente consenta di testare i

metodi tramite il seguente menu:

```
Cosa vuoi fare?
1) impostare un orario di allarme
2) attivare o disattivare un allarme
3) simulare l'avanzamento del tempo
4) uscire dal programma
```

Dopo aver eseguito ogni opzione, il programma stampa lo stato attuale della sveglia.

---

7. La classe Spostamento

Si vuole realizzare una classe che rappresenti uno spostamento nel piano. Lo spostamento è caratterizzato da due valori reali, dx e dy, che rappresentano rispettivamente lo spostamento lungo l'asse x e lo spostamento lungo l'asse y.
Due spostamenti possono essere sommati, producendo uno spostamento complessivo in cui dx è pari alla somma dei due spostamenti lungo l'asse x, e dy è pari alla somma dei due spostamenti lungo l'asse y.
E' anche possibile moltiplicare uno spostamento per un numero reale K, ottenendo così uno spostamento in cui i valori di dx e dy sono pari ai rispettivi valori originali moltiplicati per k.
Esempi
Esempio:

SpostamentoA = (dx=3 , dy=2)
SpostamentoB = (dx=5 , dy=-1)
SpostamentoC = SpostamentoA + SpostamentoB ---> SpostamentoC = (dx=3+5 , dy=2-1)
SpostamentoD = SpostamentoA * 10 ---> SpostamentoD = (dx=30 , dy=20)

Definire la classe Spostamento fornendo:

a. un costruttore che riceve come parametri dx e dy

b. il metodo **void stampa()**

c. il metodo **Spostamento somma(Spostamento& s2)** che restituisce lo spostamento complessivo ottenuto dalla somma tra lo spostamento corrente ed s2. Nota: il metodo restituisce un nuovo spostamento, mentre lo spostamento corrente rimane immutato!

d. il metodo **Spostamento moltiplicazione(float k)** che restituisce il prodotto tra lo spostamento corrente e k

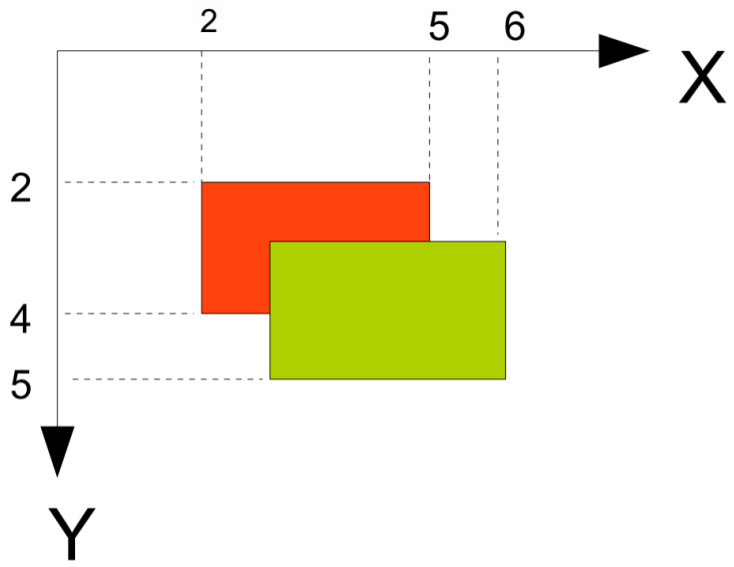
Scrivere un programma che consenta di testare i metodi. Allocare gli oggetti in modo dinamico.

---

8. Unione e intersezione di rettangoli

Si vuole creare una classe, denominata Rettangolo, che rappresenta un'area rettangolare sullo schermo. Ciascuna area è caratterizzata da:

- cordinate (x,y) del vertice "in alto a sinistra"
- altezza e larghezza del rettangolo



Supporre che gli assi siano disposti come in figura. Esempio:

Rettangolo rosso: (x=2, y=2, larghezza=3, altezza=2)
Rettangolo verde: (x=3, y=3, larghezza=3, altezza=2)

Oltre agli usuali metodi (set,get,acquisizione,stampa...) la classe Rettangolo esporrà i seguenti metodi pubblici:

a. il metodo **bool contiene(int x, int y)** che restitusice *true* se il rettangolo corrente contiene il punto (x,y)

b. il metodo **bool contiene(Punto& p)** che restitusice *true* se il rettangolo corrente contiene il punto p

c. il metodo **bool contiene(Rettangolo& r)** che restitusice *true* se il rettangolo corrente contiene completamente il rettangolo r

d. il metodo **bool interseca(Rettangolo& r)** che restitusice *true* se il rettangolo corrente interseca r. Suggerimento: due rettangoli NON si intersecano se e solo il primo rettangolo è completamente sopra, oppure completamente sotto, oppure completamente a sinistra, oppure completamente a destra del prima. Se nessuna delle quattro precedenti condizioni è vera, allora sicuramente i rettnagoli si intersecano.

e. il metodo **creaUnione** che riceve come parametro un secondo rettangolo r, e restituisce un nuovo rettangolo pari all'unione fra il rettangolo corrente ed il rettangolo r (vedi esempio).

f. il metodo **creaIntersezione** che riceve come parametro un secondo rettangolo r, e restituisce un nuovo rettangolo pari all'intersezione fra il rettangolo corrente ed il rettangolo r (Vedi esempio). Se i due rettangoli non si intersecano viene restiutito un rettangolo "nullo".

Nell'esempio, l'unione è il rettangolo caratterizzato dai seguenti valori: (x=2, y=2, larghezza=4, altezza=3)
mentre l'intersezione è il rettangolo con valori (x=3, y=3, larghezza=2, altezza=1)
Scrivere un programma che richiede dall'utente i dati di due rettangoli, e calcola i rettangoli "unione" e "intersezione".
