

main.cpp(tutto)

mercoledì 28 giugno 2023 21:32

```
/*Simulazione d'esame del 07/06/2023
*/
#include <iostream> // per usare cout e cin
#include <string> // per usare le stringhe
using namespace std;

// Funzione ausiliaria che riceve in input due stringhe che rappresentano due
date nel formato "gg/mm/aaaa" e restituisce true se la prima data è precedente
o uguale alla seconda, false altrimenti. Si assuma che le stringhe siano
sempre nel formato corretto e che le date siano valide.
bool data_precedente(string data1, string data2) {
    // Si convertono le stringhe in interi per confrontare le date
    int gg1 = stoi(data1.substr(0, 2));
    int mm1 = stoi(data1.substr(3, 2));
    int aaaa1 = stoi(data1.substr(6, 4));
    int gg2 = stoi(data2.substr(0, 2));
    int mm2 = stoi(data2.substr(3, 2));
    int aaaa2 = stoi(data2.substr(6, 4));

    // Si confrontano prima gli anni, poi i mesi e infine i giorni
    if (aaaa1 < aaaa2) {
        return true;
    } else if (aaaa1 == aaaa2) {
        if (mm1 < mm2) {
            return true;
        } else if (mm1 == mm2) {
            if (gg1 <= gg2) {
                return true;
            } else {
                return false;
            }
        } else {
            return false;
        }
    } else {
        return false;
    }
}

/*Si consideri una classe Prodotto che rappresenta un prodotto venduto in un
supermercato. La classe ha i seguenti attributi privati:
codice, nome, prezzo, scadenza
La classe ha i seguenti metodi pubblici:
costruttore, get degli attributi, stampa
*/

class Prodotto {
private:
    int codice;
    string nome;
    double prezzo;
    string scadenza;
public:
    Prodotto(int c, string n, double p, string s) {
```

```

        codice = c;
        nome = n;
        prezzo = p;
        scadenza = s;
    }
    int get_codice() {
        return codice;
    }
    string get_nome() {
        return nome;
    }
    double get_prezzo() {
        return prezzo;
    }
    string get_scadenza() {
        return scadenza;
    }
    void stampa() {
        cout << codice << ": " << nome << " - " << prezzo << " euro (scadenza: "
<< scadenza << ")" << endl;
    }
    virtual ~Prodotto() {};
};
/*

```

Si consideri una classe Sconto che rappresenta uno sconto applicabile a un prodotto.

La classe ha i seguenti attributi privati:

percentuale, scadenza

```

*/
class Sconto {
private:
    double percentuale;
    string scadenza;
public:
    Sconto(double p, string s) {
        percentuale = p;
        scadenza = s;
    }
    double get_percentuale() {
        return percentuale;
    }
    string get_scadenza() {
        return scadenza;
    }
    double applica(double prezzo) {
        return prezzo * (1 - percentuale / 100);
    }
    void stampa() {
        cout << "[Sconto del " << percentuale << "% valido fino al " << scadenza
<< "]"<<endl;
    }
};

```

/*

Si consideri una classe ProdottoScontato che deriva dalla classe Prodotto e ha un attributo privato aggiuntivo:

- sconto: un puntatore a Sconto che indica lo sconto applicabile al prodotto

La classe ha i seguenti metodi pubblici:

- get_prezzo_scontato: restituisce il prezzo del prodotto dopo aver applicato lo sconto, se lo sconto è valido rispetto alla data odierna, altrimenti restituisce il prezzo originale del prodotto

```

        - stampa: stampa a video le informazioni del prodotto nel formato
"codice: nome - prezzo euro (scadenza: gg/mm/aaaa) [sconto del percentuale%
valido fino al scadenza]", se lo sconto è valido rispetto alla data odierna,
altrimenti stampa le informazioni del prodotto senza lo sconto
*/

```

```

class ProdottoScontato : public Prodotto {
private:
    Sconto* sconto;
public:
    ProdottoScontato(int c, string n, double p, string s, Sconto* sc) :
Prodotto(c, n, p, s) {
        sconto = sc;
    }
    ~ProdottoScontato() {
        delete sconto;
    }
    Sconto* get_sconto() {
        return sconto;
    }
    double get_prezzo_scontato(string oggi) {
        if (data_precedente(oggi, sconto->get_scadenza())) {
            // Lo sconto è valido, si applica al prezzo originale
            return sconto->applica(get_prezzo());
        } else {
            // Lo sconto non è valido, si restituisce il prezzo originale
            return get_prezzo();
        }
    }
    void stampa(string oggi) {
        //if (data_precedente(sconto->get_scadenza(), oggi)) {
            // Lo sconto è valido, si stampa con le informazioni dello sconto
            Prodotto::stampa();
            sconto->stampa();
        //} else {
            // Lo sconto non è valido, si stampa senza lo sconto
            // Prodotto::stampa();
        //}
    }
};
// Si consideri una classe Pila che rappresenta una pila di prodotti.

```

```

class Pila {
private:
    Prodotto** dati;
    int dim;
    int testa;
public:
    Pila(int d) {
        dim = d;
        testa = -1;
        dati = new Prodotto*[dim];
    }
    ~Pila() {
        while (!is_empty()) {
            delete pop();
        }
        delete[] dati;
    }
    void push(Prodotto* p) {
        if (!is_full()) {
            testa++;
            dati[testa] = p;
        }
    }

```

```

    } else {
        cout << "Pila piena, impossibile inserire il prodotto" << endl;
    }
}
Prodotto* pop() {
    if (!is_empty()) {
        Prodotto* p = dati[testa];
        testa--;
        return p;
    } else {
        cout << "Pila vuota, impossibile rimuovere il prodotto" << endl;
        return nullptr;
    }
}
Prodotto* top() {
    if (!is_empty()) {
        return dati[testa];
    } else {
        cout << "Pila vuota, impossibile visualizzare il prodotto" << endl;
        return nullptr;
    }
}
bool is_empty() {
    return testa == -1;
}
bool is_full() {
    return testa == dim - 1;
}
};

```

// Si scriva un programma principale che crea una pila di prodotti di dimensione 10 e inserisce nella pila i seguenti prodotti:
// - codice 1, nome "latte", prezzo 1.2, scadenza "10/06/2023"
// - codice 2, nome "pane", prezzo 0.8, scadenza "08/06/2023"
// - codice 3, nome "burro", prezzo 2.5, scadenza "15/06/2023"
// - codice 4, nome "marmellata", prezzo 3.0, scadenza "31/12/2023"
// - codice 5, nome "yogurt", prezzo 1.5, scadenza "12/06/2023", sconto del 20% valido fino al "10/06/2023"
// - codice 6, nome "succo di frutta", prezzo 2.0, scadenza "30/06/2023", sconto del 10% valido fino al "15/06/2023"
// Il programma poi chiede all'utente di inserire la data odierna nel formato "gg/mm/aaaa"
// e controlla se ci sono prodotti nella pila che sono scaduti rispetto a tale data. In tal caso, il programma deve rimuovere dalla pila i prodotti scaduti e stampare un messaggio di avviso con il nome e il codice dei prodotti rimossi.
// Il programma poi stampa il contenuto della pila e calcola il totale da pagare per i prodotti nella pila usando i prezzi scontati dei prodotti scontati, se lo sconto è valido rispetto alla data odierna, altrimenti usando i prezzi originali.

```

int main() {
    Pila p(10);
    p.push(new Prodotto(1, "latte", 1.2, "10/06/2023"));
    p.push(new Prodotto(2, "pane", 0.8, "08/06/2023"));
    p.push(new Prodotto(3, "burro", 2.5, "15/06/2023"));
    p.push(new Prodotto(4, "marmellata", 3.0, "31/12/2023"));
    p.push(new ProdottoScontato(5, "yogurt", 1.5, "12/06/2023", new Sconto(20, "10/06/2023")));
    p.push(new ProdottoScontato(6, "succo di frutta", 2.0, "30/06/2023", new Sconto(10, "15/06/2023")));

    cout << "Inserisci la data odierna (gg/mm/aaaa): ";
}

```

```

string oggi;
cin >> oggi;

// Si crea una pila ausiliaria per conservare i prodotti non scaduti
Pila q(10);

// Si scorre la pila originale e si controlla se ci sono prodotti scaduti
while (!p.is_empty()) {
    Prodotto* prod = p.pop();
    if (data_precedente(prod->get_scadenza(), oggi)) {
        // Il prodotto è scaduto, lo si rimuove e si stampa un messaggio
        cout << "Attenzione: il prodotto " << prod->get_nome() << " (codice: "
<< prod->get_codice() << ") è scaduto e verrà rimosso dalla pila" << endl;
        delete prod;
    } else {
        // Il prodotto non è scaduto, lo si inserisce nella pila ausiliaria
        q.push(prod);
    }
}

cout << "Contenuto della pila:" << endl;
double totale = 0;

// Si riportano i prodotti non scaduti nella pila originale e si calcola il
totale
while (!q.is_empty()) {
    Prodotto* prod = q.pop();
    p.push(prod);
    //prod->stampa();
    // Si controlla se il prodotto è scontato e se lo sconto è valido
    ProdottoScontato* prods = dynamic_cast<ProdottoScontato*>(prod);
    if (prods != nullptr && data_precedente(oggi, prods->get_scadenza())) {
        // Si usa il prezzo scontato
        totale += prods->get_prezzo_scontato(oggi);
        prods->stampa(oggi);
    } else {
        // Si usa il prezzo originale
        totale += prod->get_prezzo();
        prod->stampa();
    }
}

cout << "Totale da pagare: " << totale << " euro" << endl;

return 0;
}

```

Simulazione d'esame del 07/06/2023

Durata: 2 ore

Si consideri una classe Prodotto che rappresenta un prodotto venduto in un supermercato. La classe ha i seguenti attributi privati:

codice, nome, prezzo, scadenza

La classe ha i seguenti metodi pubblici:

costruttore, get degli attributi, stampa

Si consideri una classe Sconto che rappresenta uno sconto applicabile a un prodotto. La classe ha i seguenti attributi privati:

percentuale, scadenza

Si consideri una classe ProdottoScontato che deriva dalla classe Prodotto e ha un attributo privato aggiuntivo:

- *sconto: un puntatore a Sconto che indica lo sconto applicabile al prodotto*

La classe ha i seguenti metodi pubblici:

- *get_prezzo_scontato*: restituisce il prezzo del prodotto dopo aver applicato lo sconto, se lo sconto è valido rispetto alla data odierna, altrimenti restituisce il prezzo originale del prodotto
- *stampa*: stampa a video le informazioni del prodotto nel formato "codice: nome - prezzo euro (scadenza: gg/mm/aaaa) [sconto del percentuale% valido fino al scadenza]", se lo sconto è valido rispetto alla data odierna, altrimenti stampa le informazioni del prodotto senza lo sconto

Si scriva un programma che crea una pila di prodotti di dimensione 10 e inserisce nella pila i seguenti prodotti:

- *codice 1, nome "latte", prezzo 1.2, scadenza "10/06/2023"*
- *codice 2, nome "pane", prezzo 0.8, scadenza "08/06/2023"*
- *codice 3, nome "burro", prezzo 2.5, scadenza "15/06/2023"*
- *codice 4, nome "marmellata", prezzo 3.0, scadenza "31/12/2023"*
- *codice 5, nome "yogurt", prezzo 1.5, scadenza "12/06/2023", sconto del 20% valido fino al "10/06/2023"*
- *codice 6, nome "succo di frutta", prezzo 2.0, scadenza "30/06/2023", sconto del 10% valido fino al "15/06/2023"*

Il programma poi chiede all'utente di inserire la data odierna nel formato "gg/mm/aaaa" e controlla se ci sono prodotti nella pila che sono scaduti rispetto a tale data. In tal caso, il programma deve rimuovere dalla pila i prodotti scaduti e stampare un messaggio di avviso con il nome e il codice dei prodotti rimossi.

Il programma poi stampa il contenuto della pila e calcola il totale da pagare per i prodotti nella pila usando i prezzi scontati dei prodotti scontati, se lo sconto è valido rispetto alla data odierna, altrimenti usando i prezzi originali.

Output atteso inserendo la data odierna (07/06/2023):

Inserisci la data odierna (gg/mm/aaaa): 07/06/2023

Contenuto della pila:

1: latte - 1.2 euro (scadenza: 10/06/2023)

2: pane - 0.8 euro (scadenza: 08/06/2023)

3: burro - 2.5 euro (scadenza: 15/06/2023)

4: marmellata - 3 euro (scadenza: 31/12/2023)

5: yogurt - 1.5 euro (scadenza: 12/06/2023)

[Sconto del 20% valido fino al 10/06/2023]

6: succo di frutta - 2 euro (scadenza: 30/06/2023)

[Sconto del 10% valido fino al 15/06/2023]

Totale da pagare: 10.5 euro

Ritaglio schermata acquisito: 28/06/2023 21:37