

# Veicolo.h

mercoledì 28 giugno 2023 22:03

```
#include <iostream>
#include <string>
using namespace std;
#ifndef VEICOLO_H
#define VEICOLO_H
class Veicolo{
    private:
        string modello;
        string marca;
        int anno;
    public:
        Veicolo(string _modello, string _marca, int _anno):
            modello(_modello), marca(_marca), anno(_anno){};
        virtual ~Veicolo(){};
        string getModello() const{return this->modello;}
        string getMarca() const {return this->marca;}
        int getAnno() const{return this->anno;}
        virtual void stampa() const;
        virtual string tipo() const = 0;
        friend ostream &operator <<(ostream &os, const Veicolo& v);
    };
    ostream &operator <<(ostream &os, const Veicolo& v){
        os << "Modello: " << v.getModello() << ", Marca: " << v.getMarca()
        << ", Anno:" << v.getAnno() << ", Tipo:" << v.tipo() << endl;
        return os;
    }
    void Veicolo::stampa() const{
        cout << "Modello: " << getModello() << ", Marca: " << getMarca()
        << ", Anno:" << getAnno() << ", Tipo: " << tipo() << endl;
    }
    class Auto : public Veicolo{
        public:
            Auto(string _modello, string _marca, int _anno) :
                Veicolo(_modello, _marca, _anno){};
            ~Auto(){}
            string tipo() const{ return "auto";}
    };
    class Moto : public Veicolo{
        public:
            Moto(string _modello, string _marca, int _anno) :
                Veicolo(_modello, _marca, _anno){};
            ~Moto(){}
            string tipo() const{ return "moto";}
    };
}
#endif
```

# Coda.h template

mercoledì 28 giugno 2023 22:03

```
#include <iostream>
#include "Veicolo.h"
using namespace std;
template <typename T = Veicolo*>
class Nodo{
public:
    Nodo<T>* succ;
    T dato;
    Nodo(T el) : dato(el), succ(nullptr){};
};

template <typename T = Veicolo*>
class Coda{
private:
    Nodo<T>* testa;
    Nodo<T>* fine;
    int num_el = 0;
public:
    Coda() : testa(nullptr), fine(nullptr){};
    ~Coda();
    bool isEmpty();
    Nodo<T>* getTesta() const{return this->testa;}
    void enqueue(T el);
    T dequeue();
    void stampaCoda() const;
    int dimensione() const{return this->num_el;}
    T front() const;
};
template <typename T>
bool Coda<T>::isEmpty(){
    if(fine == nullptr){
        return true;
    }
    else{
        return false;
    }
}

template <typename T>
Coda<T>::~Coda(){
    while(!isEmpty()){
        dequeue();
    }
}
template <typename T>
void Coda<T>::enqueue(T el){
    Nodo<T>* nuovo = new Nodo<T>(el);
    num_el++;
    if(fine == nullptr){
        testa = fine = nuovo;
    }
    else{
        fine->succ = nuovo;
        fine = nuovo;
    }
}
```

```

}

template <typename T>
T Coda<T>::dequeue(){
    if(isEmpty()){
        throw runtime_error("la coda e' vuota!");
    }
    T aux = testa->dato;
    Nodo<T>* iter = testa;
    testa = testa->succ;
    if(testa == nullptr){
        fine = nullptr;
    }
    delete iter;
    num_el--;
    return aux;
}
template <typename T>
T Coda<T>::front() const{
    if(isEmpty()){
        throw runtime_error("la coda e' vuota");
    }
    return testa->dato;
}
template <typename T>
void Coda<T>::stampaCoda() const{
    Nodo<T>* iter = testa;

    while(iter != nullptr){
        cout << *iter->dato;
        iter = iter->succ;
    }
}

```

# main.cpp

mercoledì 28 giugno 2023 22:04

```
#include <iostream>
#include <string>
#include "Veicolo.h"
#include "CodaT.h"
int rimuovi(string &risp, Coda<Veicolo*> &tail){
    Nodo<Veicolo*> *iter;
    Coda tmp;
    string tmp2;
    int count = 0;
    for(iter = tail.getTesta(); iter != nullptr; iter = iter->succ){
        tmp2 = iter->dato->tipo();
        if(tmp2 != risp){
            tmp.enqueue(iter->dato);
        }
        else{
            count++;
        }
    }
    while(!tail.isEmpty()){
        tail.dequeue();
    }
    while(!tmp.isEmpty()){
        tail.enqueue(tmp.dequeue());
    }
    return count;
}

int main(){
    Veicolo* a1 = new Auto("Fiesta", "Ford", 2010);
    Veicolo* m1 = new Moto("CBR 600 RR", "Honda", 2008);
    Veicolo* a2 = new Auto("Panda", "Fiat", 2012);
    Veicolo* m2 = new Moto("Ninja ZX-10R", "Kawasaki", 2019);
    Veicolo* a3 = new Auto("Golf", "Volkswagen", 2015);
    Veicolo* m3 = new Moto("R1", "Yamaha", 2020);
    Coda tail;
    tail.enqueue(a1);
    tail.enqueue(m1);
    tail.enqueue(a2);
    tail.enqueue(m2);
    tail.enqueue(a3);
    tail.enqueue(m3);

    cout << "STAMPA CODA: \n";
    tail.stampaCoda();

    string risp;
    cout << "che tipo di veicolo vuoi rimuovere?(auto/moto): ";
    cin >> risp;
    rimuovi(risp, tail);
    cout << "\n\nSTAMPA DOPO RIMOZIONE: \n";
    tail.stampaCoda();

    delete a1;
    delete m1;
    delete a2;
```

```
    delete m2;
    delete a3;
    delete m3;
    return 0;
}
```

# Testo

mercoledì 28 giugno 2023 22:01

---

Si consideri una classe virtuale Veicolo che rappresenta un veicolo.

La classe ha i seguenti attributi privati:

- *modello, marca, anno*

La classe ha i seguenti metodi pubblici:

- *costruttore, get degli attributi, stampa*
- *un metodo virtuale puro chiamato tipo che restituisce il tipo di veicolo*

Si considerino due classi derivate da Veicolo: Auto e Moto. Ognuna di queste classi implementa il metodo tipo in modo appropriato per il tipo di veicolo.

Si consideri una classe template Coda<T> che rappresenta una struttura dati a coda di elementi di tipo T.

La classe ha i seguenti attributi privati:

- *testa: un puntatore al nodo in testa alla coda*
- *coda: un puntatore al nodo in coda alla coda*
- *dimensione: il numero di nodi nella coda*

La classe ha i seguenti metodi pubblici:

- *costruttore, distruttore*
- *enqueue: inserisce un nuovo elemento in coda alla coda*
- *dequeue: rimuove e restituisce l'elemento in testa alla coda, se la coda non è vuota*
- *front: restituisce l'elemento in testa alla coda senza rimuoverlo, se la coda non è vuota*
- *stamp: stampa tutti gli elementi nella coda dal primo all'ultimo inserito*

Si scriva un programma che crea una coda di puntatori a veicoli e inserisce nella coda i seguenti veicoli usando il metodo enqueue:

- *Fiesta, Ford, 2010, tipo Auto*
- *CBR 600 RR, Honda, 2008, tipo Moto*
- *Panda, Fiat, 2012, tipo Auto*
- *Ninja ZX-10R, Kawasaki, 2019, tipo Moto*
- *Golf, Volkswagen, 2015, tipo Auto*
- *R1, Yamaha, 2020, tipo Moto*

Il programma poi chiede all'utente di inserire il tipo di veicoli da rimuovere dalla coda e li rimuove usando il metodo *dequeue* della classe Coda. Il programma poi stampa tutti i veicoli rimanenti nella coda usando il metodo stampa della classe Coda. Per rimuovere i veicoli di un certo tipo dalla

coda si scriva una funzione ricorsiva che riceve come parametri la coda e il tipo da rimuovere e restituisce il numero di veicoli rimossi.

**Output atteso inserendo il tipo Auto:**

*Inserisci il tipo di veicoli da rimuovere: Auto*

*Sono stati rimossi 3 veicoli dalla coda.*

*Veicoli rimanenti nella coda:*

*CBR 600 RR - Honda - 2008 - Moto*

*Ninja ZX-10R - Kawasaki - 2019 - Moto*

*R1 - Yamaha - 2020 - Moto*

**Output atteso inserendo il tipo Moto:**

*Inserisci il tipo di veicoli da rimuovere: Moto*

*Sono stati rimossi 3 veicoli dalla coda.*

*Veicoli rimanenti nella coda:*

*Fiesta - Ford - 2010 - Auto*

*Panda - Fiat - 2012 - Auto*

*Golf - Volkswagen - 2015 - Auto*