Veicoli.h

mercoledì 28 giugno 2023 21:46

```
#include <iostream>
#include <string>

#ifndef VEICOLI_H
#define VEICOLI_H
class Veicolo{
    private:
        int targa;
        std::string nome_modello;
        int anno_fabbricazione;
        int cilindrata;
        std::string marca;
    public:

        Veicolo(int t, std::string nome_m, int anno_f, int cil, std::string
m) :
            targa(t), nome_modello(nome_m), anno_fabbricazione(anno_f),
cilindrata(cil), marca(m){};
        ~Veicolo(){};


        int getTarga() const{ return this->targa;}
        std::string getNomeModello() const{ return this->nome_modello;}
        int getAnnoFabbricazione() const{return this->anno_fabbricazione;}
        int getCilindrata() const{ return this->cilindrata;}
        std::string getMarca() const{ return this->marca;}
        virtual void stampa() = 0;
        Veicolo(Veicolo&v);

};
std::ostream& operator<<(std::ostream& os, const Veicolo& a) {
    os << "Veicolo: " << a.getTarga() << ", Modello:" << a.getNomeModello()
<< std::endl;
    os << "Marca: " << a.getMarca() << ", Anno Fabbricazione: "
<< a.getAnnoFabbricazione() << std::endl;
    os << "Cilindrata: " << a.getCilindrata();
    return os;
}


Veicolo::Veicolo(Veicolo& v){
    this->targa = v.targa;
    this->nome_modello = v.nome_modello;
    this->anno_fabbricazione = v.anno_fabbricazione;
    this->cilindrata = v.cilindrata;
    this->marca = v.marca;
}
class Auto : public Veicolo{
    private:
        int numero_porte;
    public:
        Auto(int t, std::string nome_m, int anno_f, int cil, std::string m,
int num_p) :
            Veicolo(t, nome_m, anno_f, cil, m), numero_porte(num_p){};
        Auto(Auto& a);
```

```
        ~Auto(){};

        int getNumeroPorte() const{ return this->numero_porte;}
        void stampa();
};
Auto::Auto(Auto& a) : Veicolo(a){
    this->numero_porte = numero_porte;
}

std::ostream& operator<<(std::ostream& os, const Auto& a) {
    os << "Auto: " << a.getTarga() << ", Modello:" << a.getNomeModello()
<< std::endl;
    os << "Marca: " << a.getMarca() << ", Anno Fabbricazione: "
<< a.getAnnoFabbricazione() << std::endl;
    os << "Cilindrata: " << a.getCilindrata() << "Numero porte: "
<< a.getNumeroPorte() << std::endl;
    return os;
}

void Auto::stampa()
{
    std::cout << "Veicolo: " << std::endl;
    std::cout << "targa: " << getTarga() << "\t nome_modello: "
<< getNomeModello()
    << "\n anno fabbricazione: " << getAnnoFabbricazione() << "\t cilindrata:
" << getCilindrata()
    << "\n marca: " << getMarca() << "\t numero porte: "
<< getNumeroPorte()  << std::endl;
    std::cout << std::endl;
}
class Moto : public Veicolo{
    private:
        std::string tipo_moto;
    public:
        Moto(int t, std::string nome_m, int anno_f, int cil, std::string m,
std::string tipo_m)
            : Veicolo(t, nome_m, anno_f, cil, m), tipo_moto(tipo_m){};
        ~Moto(){};
        std::string getTipoMoto() const{ return this->tipo_moto;}
        void stampa();
};
std::ostream& operator<<(std::ostream& os, const Moto& a) {
    os << "Auto: " << a.getTarga() << ", Modello:" << a.getNomeModello()
<< std::endl;
    os << "Marca: " << a.getMarca() << ", Anno Fabbricazione: "
<< a.getAnnoFabbricazione() << std::endl;
    os << "Cilindrata: " << a.getCilindrata() << "Tipo moto: "
<< a.getTipoMoto() << std::endl;
    return os;
}


void Moto::stampa()
{
    std::cout << "Veicolo: " << std::endl;
    std::cout << "targa: " << getTarga() << "\t nome_modello: "
<< getNomeModello()
    << "\n anno fabbricazione: " << getAnnoFabbricazione() << "\t cilindrata:
" << getCilindrata()
    << "\n marca: " << getMarca() << "\t tipo moto: " << getTipoMoto()
<< std::endl;
```

```
    std::cout << std::endl;
}
#endif
```

BST.h

mercoledì 28 giugno 2023 21:49

```
#include <iostream>
#include "veicoli.h"
using namespace std;
#ifndef BST_H
#define BST_H
class Nodo{
    public:
        Nodo* left;
        Nodo* right;
        Nodo* padre;
        Veicolo* dato;
        Nodo(Veicolo* v) : dato(v), left(nullptr), right(nullptr),
padre(nullptr){};
};
class BST{
    private:
        Nodo* radice;
        void inOrder(Nodo* ptr) const;
        void inOrder(Nodo* ptr, ostream& os) const;
        void preOrder(Nodo* ptr) const;
        void trapianta(Nodo*u, Nodo* v);
        void cancella(Nodo* z);
        Nodo* minimo(Nodo* x);
        void distruggi(Nodo* nodo);
        Nodo* ricerca(Nodo* x, int el) const;

    public:
        BST() : radice(nullptr){};
        ~BST();
        Nodo* getRadice() const{ return this->radice;}
        void inserisci(Veicolo* el);
        void inOrder() const;
        void preOrder() const;
        friend ostream &operator << (ostream &os, const BST &a);
        bool cancella(int x);
        Nodo* ricerca(int targa_) const;

};
void BST::inserisci(Veicolo* el){ //i veicoli verranno inseriti in funzione
del valore della loro targa.
    Nodo* y = nullptr;
    Nodo *nuovo = new Nodo(el);
    nuovo->left = nullptr;
    nuovo->right = nullptr;
    Nodo *x = radice;
    while(x != nullptr){
        y = x;
        if(el->getTarga() <  x->dato->getTarga()){
            x = x->left;
        }
        else
            x = x->right;
    }
    nuovo->padre = y;
    if(y == nullptr){
```

```
            radice = nuovo;
        }
        else if(el->getTarga() < y->dato->getTarga()){
            y->left = nuovo;
        }
        else{
            y->right = nuovo;
        }
}
void BST::inOrder(Nodo* ptr) const{
    if(ptr != nullptr){
        inOrder(ptr->left);
        ptr->dato->stampa();
        // cout << *(ptr->dato) << endl;
        inOrder(ptr->right);
    }
}
void BST::inOrder() const{
    inOrder(radice);
}
void BST::inOrder(Nodo* ptr, ostream& os) const { //stampa in ordine
alfabetico perchè i veicoli sono inseriti nell'albero in funzione della loro
targa

    if (ptr != nullptr) {
        inOrder(ptr->left, os);
        os << *(ptr->dato) << endl;
        inOrder(ptr->right, os);
    }
}

ostream &operator << (ostream &os, const BST &a){
    a.inOrder(a.radice, os);
    return os;
}
void BST::trapianta(Nodo*u, Nodo* v){
    if(u->padre == nullptr){
        radice = v;
    }
    else if(u->padre->left == u){
        u->padre->left = v;
    }
    else
        u->padre->right = v;
}
Nodo* BST::minimo(Nodo* x){
    Nodo* min = x;
    while(min->left != nullptr)
        min = min->left;
    return min;
}
void BST::cancella(Nodo *z){
    Nodo* y;
    if(z->left == nullptr){
        trapianta(z, z->right);
    }
    else if(z->right == nullptr){
        trapianta(z, z->left);
    }
    else{
        y = minimo(z->right); //y puntera' al successore di z
        if(y->padre != z){
```

```
            trapianta(y, y->right);
            y->right = z->right;
            y->right->padre = y;
        }
        trapianta(z, y);
        y->left = z->left;
        y->left->padre = y;
    }
    delete z;
}
bool BST::cancella(int x){
    Nodo* iter = radice;
    while((iter != nullptr) && (x != iter->dato->getTarga())){
        if( x < iter->dato->getTarga())
            iter = iter->left;
        else
            iter = iter->right;
    }
    if(iter == nullptr){
        return false;
    }
    else{
        cancella(iter);
        return true;
    }
}
void BST::preOrder(Nodo* ptr) const{
    if(ptr != nullptr){
        ptr->dato->stampa();
        preOrder(ptr->left);
        preOrder(ptr->right);
    }
}
void BST::preOrder() const{
    preOrder(radice);
}
BST::~BST() {
    distruggi(radice);
}
void BST::distruggi(Nodo* nodo) {
    if (nodo != nullptr) {
        distruggi(nodo->left);  // distruggi sottoalbero sinistro
        distruggi(nodo->right); // distruggi sottoalbero destro
        delete nodo->dato;      // distruggi il dato del nodo
        delete nodo;            // distruggi il nodo stesso
    }
}

 Nodo* BST::ricerca(Nodo* x, int el) const{
    if( x == nullptr || x->dato->getTarga() == el){
        return x;
    }
    if(el < x->dato->getTarga())
        return ricerca(x->left, el);
    else
        return ricerca(x->right, el);
}
 Nodo* BST::ricerca(int targa) const{
    return ricerca(radice, targa);
 }
#endif
```

main.cpp

mercoledì 28 giugno 2023 21:49

```
#include <iostream>
#include "veicoli.h"
#include "bst.h"
int main(){
    Veicolo *a = new Auto( 1, "Panda FIRE", 1986, 769, "Fiat", 3);
    Veicolo *a2 = new Auto(3, "Ferrari Testarossa", 1984, 4943, "Ferrari", 3);
    Veicolo *m = new Moto(4, "YZF-R3", 2014, 249, "Yamaha", "Sportiva" );
    Veicolo *a3 = new Auto(5, "Peugeot 208", 2019, 1199, "Peugeto", 5);
    Veicolo *m2 = new Moto(6,"Honda AfricaTwin 650", 1983, 749,"Honda",
"Enduro");
    // a->stampa();
    // m->stampa();

    BST *tree = new BST;;
    tree->inserisci(a2);
    tree->inserisci(a);
    tree->inserisci(a3);
    tree->inserisci(m);
    tree->inserisci(m2);
    // tree.inOrder();
    tree->cancella(3);
    tree->cancella(5);
    tree->preOrder();
    delete tree;
    return 0;
}
```

Testo

mercoledì 28 giugno 2023 21:50

Simulazione d'esame

Si consideri una classe astratta Veicolo. La classe ha i seguenti attributi privati:

- *Targa, nome modello, anno di fabbricazione, marca.*

La classe ha i seguenti metodi pubblici:

- *costruttore, get degli attributi*
- *un metodo virtuale puro per la stampa*

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

La classe Auto ha i seguenti attributi privati:

- *numero_porte*

La classe Moto ha i seguenti attributi privati:

- *tipo_moto (sportiva, chopper, enduro, ecc)*

Si consideri una classe AlberoBinario che rappresenta una struttura dati ad albero binario di ricerca di veicoli. La classe ha i seguenti attributi privati:

- *radice: un puntatore al nodo radice dell'albero*

La classe ha i seguenti metodi pubblici:

- *costruttore, distruttore*
- *inserisci: inserisce un nuovo veicolo nell'albero in base alla sua targa*
- *elimina: elimina un nuovo veicolo nell'albero in base alla sua targa*
- *cerca: cerca un veicolo nell'albero data la sua targa e restituisce un puntatore al veicolo se trovato, altrimenti restituisce nullptr*

trovato, altrimenti restituisce nullptr

- *stampa: stampa tutti i veicoli nell'albero in ordine alfabetico delle loro targe*
- *verifica stampa: stampa tutti i veicoli eseguendo una visita Pre-Order dell'albero*

Si scriva un programma che crei un albero binario di veicoli e inserisce nell'albero i seguenti veicoli:

Auto:
Targa: 1 Nome modello: Panda FIRE
Marca: Fiat Anno fabbricazione: 1986
Cilindrata: 769 Numero porte: 3

Auto:
Targa: 3 Nome modello: Ferrari Testarossa
Marca: Ferrari Anno fabbricazione: 1984
Cilindrata: 4943 Numero porte: 3
Moto:
Targa: 4 Nome modello: YZF-R3
Marca: Yamaha Anno fabbricazione: 2014
Cilindrata: 249 Tipo moto: Sportiva
Auto:
Targa: 5 Nome modello: Peugeot 208
Marca: Peugeot Anno fabbricazione: 2019
Cilindrata: 1199 Numero porte: 5
Moto:
Targa: 6 Nome modello: Honda Africa Twin 650
Marca: Honda Anno fabbricazione: 1983
Cilindrata: 749 Tipo moto: Enduro

Il programma dovrà eseguire in ordine:

- Si inseriscono i veicoli (si specifica la targa) nel seguente ordine: 3, 1, 5, 4, 6
- Si eliminano in ordine i veicoli con targa 3 e 5.
- Si stampa con una visita Pre-Order

La stampa risultate dovrà essere la seguente:

```
Moto:
```

La stampa risultate dovrà essere la seguente:

```
Moto:
    Targa: 4    Nome modello: YZF-R3
    Marca: Yamaha   Anno fabbricazione: 2014
    Cilindrata: 249 Tipo moto: Sportiva
Auto:
    Targa: 1    Nome modello: Panda FIRE
    Marca: Fiat Anno fabbricazione: 1986
    Cilindrata: 769 Numero porte: 3
Moto:
    Targa: 6    Nome modello: Honda Africa Twin 650
    Marca: Honda    Anno fabbricazione: 1983
    Cilindrata: 749 Tipo moto: Enduro
```