МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №3**

**з дисципліни «Об’єктно-орієнтоване програмування»**

**на тему: «Операції перетворення»**

Виконав: студент гр. ПЗ2011

Кулик Сергій Вадимович

Прийняла:

Демидович Інна Миколаївна

Дніпро, 2021

**Тема:** Операції перетворення

**Мета:** Навчитися працювати із операціями перетворення.

**Завдання**

Написати програму, що реалізує операції перетворення.

**Індивідуальне завдання**

|  |  |  |
| --- | --- | --- |
| Варіант | Клас | Перетворення |
| 1 | Автомобіль | char\*, int, double int, double повертають семантично різні значення |

**Опис класу**

Клас: автомобіль.

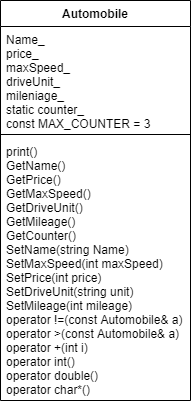
Приватні поля:

* Name\_ ­– ім’я автомобіля.
* price\_ – ціна автомобіля.
* maxSpeed\_ – максимальна швидкість.
* driveUnit\_ – привід автомобіля.
* mileniage – пробіг автомобіля
* static counter – лічильник об’єктів класу;
* MAX\_COUNTER = 3 – максимальна кількість об’єктів класу.

Публічні методи:

* GetName() – дозволяє отримати ім’я автомобіля.
* GetPrice() – дозволяє отримати ціну автомобіля.
* GetMaxSpeed() – дозволяє отримати макс.швидкість автомобіля.
* GetDriveUnit() – дозволяє отримати тип приводу автомобіля.
* GetCounter() – дозволяэ отримати кількість об’єктів класу
* SetName(string Name) – дозволяє установити ім’я автомобіля.
* SetMaxSpeed(int maxSpeed) – дозволяє установити макс.швидкість автомобіля.
* SetPrice(int price) – дозволяє установити ціну автомобіля.
* SetDriveUnit(string unit) – дозволяє установити тип приводу автомобіля.
* print() – виводить інформацію про всі поля класу.
* operator !=(const Automobile& a) – не дорівнює за назвами
* operator >(const Automobile& a) – більше за «пробігом»
* operator +(int i) – додавання «пробігу»
* operator char\*() – перевантажений оператор перетворення
* operator int() – перевантажений оператор перетворення
* operator double() – перевантажений оператор перетворення

**Діаграмма класів**

****

**Тести до програми**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Назва тесту** | **Вхідні дані** | **Вихідні дані** |
| 1 | Перетворення типів | Automobile a(“Honda”, 260, 5000, “full”, 0); | Conver to char: Toyota  Conver to int: 5000  Conver to double: 260 |

**Текст програми**

#include <iostream>

#include<cstring>

#pragma warning(disable : 4996)

using namespace std;

class Automobile

{

public:

//Конструкторы и деструктор

Automobile(string Name\_, int price\_, double maxSpeed\_, string driveUnit\_, int mileage\_);

Automobile();

~Automobile();

Automobile(const Automobile& car);

//Геттеры и сеттеры

string GetName() { return Name\_; }

int GetPrice() { return price\_; }

double GetMaxSpeed() { return maxSpeed\_; }

string GetDriveUnit() { return driveUnit\_; }

int GetMileage() { return mileage\_; }

static int GetCounter() { return counter; }

void SetName(string Name) { Name\_ = Name; }

void SetMaxSpeed(int maxSpeed) { maxSpeed\_ = maxSpeed; }

void SetPrice(int price) { price\_ = price; }

void SetDriveUnit(string unit) { driveUnit\_ = unit; }

void SetMileage(int mileage) { mileage\_ = mileage; }

//Перегруженные операторы

bool operator !=(const Automobile& a);

bool operator >(const Automobile& a);

Automobile operator +(int i);

//методы

void print();

operator char\* ()

{

char\* str = new char[30];

strcpy(str, Name\_.c\_str());

return str;

}

operator int()

{

return price\_;

}

operator double()

{

return maxSpeed\_;

}

private:

string Name\_;

int price\_;

double maxSpeed\_;

string driveUnit\_;

int mileage\_;

static int counter;

const int MAX\_COUNTER = 3;

};

int Automobile::counter = 0;

Automobile::Automobile()

{

if (counter >= MAX\_COUNTER)

{

cout << "Нельзя создать больше " << MAX\_COUNTER << " объектов!" << endl;

return;

}

counter++;

Name\_ = "Toyota";

price\_ = 5000;

maxSpeed\_ = 260;

driveUnit\_ = "Full";

mileage\_ = 0;

}

Automobile::Automobile(string Name\_, int price\_, double maxSpeed\_, string driveUnit\_, int mileage\_)

{

if (counter > MAX\_COUNTER)

{

cout << "Нельзя создать больше " << MAX\_COUNTER << " объектов!" << endl;

return;

}

counter++;

this->Name\_ = Name\_;

this->price\_ = price\_;

this->maxSpeed\_ = maxSpeed\_;

this->driveUnit\_ = driveUnit\_;

this->mileage\_ = mileage\_;

}

Automobile::Automobile(const Automobile& car)

{

if (counter > MAX\_COUNTER)

{

cout << "Нельзя создать больше " << MAX\_COUNTER << " объектов!" << endl;

return;

}

Name\_ = car.Name\_;

price\_ = car.price\_;

maxSpeed\_ = car.maxSpeed\_;

driveUnit\_ = car.driveUnit\_;

mileage\_ = car.mileage\_;

}

Automobile::~Automobile()

{

counter--;

}

void Automobile::print()

{

cout << "Название автомобиля: " << Name\_ << endl;

cout << "Цена: " << price\_ << endl;

cout << "Максимальная скорость: " << maxSpeed\_ << endl;

cout << "Пробег: " << mileage\_ << endl;

cout << "Привод: " << driveUnit\_ << endl << endl;

}

bool Automobile::operator !=(const Automobile& a)

{

if (Name\_ != a.Name\_)

{

return true;

}

else

{

return false;

}

}

bool Automobile::operator >(const Automobile& a)

{

if (mileage\_ > a.mileage\_)

{

return true;

}

else

{

return false;

}

}

Automobile Automobile::operator +(int i)

{

mileage\_ += i;

return \*this;

}

int main()

{

setlocale(LC\_ALL, "ru");

Automobile a;

int toInt;

double toDouble;

char\* toChar;

toChar = a;

toInt = a;

toDouble = a;

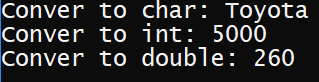
cout << "Conver to char: " << toChar << endl;

cout << "Conver to int: " << toInt << endl;

cout << "Conver to double: " << toDouble << endl;

}

**Результати**



**Висновок**

Під час виконання лабораторної роботи познайомився з перетворенням складних типів даних до простих. Операція перетворення являє собою перевантажену операцію, ім'я якої є ім'ям типу даних в який треба перетворити дані.