Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра “Електронних обчислювальних машин”



**Звіт з лабораторної роботи №5**

на тему:

“ПЕРЕВАНТАЖЕННЯ ОПЕРАТОРІВ”

Варіант 13

**Виконав:**

Ст. гр. КІ-15

Когутич В.М.

**Перевірив:**

Викладач

Козак Н.Б.

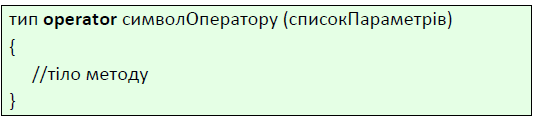
Львів – 2020

**Мета роботи:** познайомитися із перевантаженням операторів.

**Теоретичні відомості:**

**Перевантаження операторів**

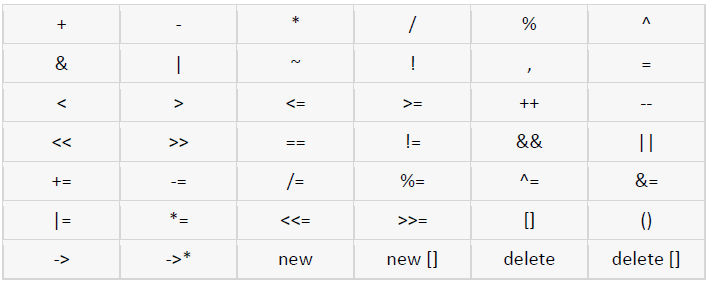
Кожному оператору мова С++ ставить у відповідність ім'я функції, що складається з ключового слова operator, власне оператору та аргументів відповідних типів:



Щоб використовувати операцію над об'єктами класів, ця операція повинна бути перевантажена, але є два виключення. Операції присвоювання (=) і взяття адреси (&) створюються в класі автоматично за замовчуванням, тому їх можна використовувати без явного перевантаження. За замовчуванням операція присвоювання зводиться до побітового копіювання даних-елементів класу. Проте таке побітове копіювання небезпечне для класів з елементами, що вказують на динамічно виділені області пам'яті, масиви, рядки, оскільки в цьому випадку відбувається копіювання не даних (глибоке копіювання), а лише вказівників на дані (поверхневе копіювання). Для таких класів слід явно перевантажувати операцію присвоювання і здійснювати у ній глибоке копіювання. Операція адресації також може бути використана з об'єктами будь-яких класів без перевантаження. Вона просто повертає адресу об'єкта в пам'яті. Але операцію адресації можна також і перевантажувати.

При перевантаженні операцій ( ), [], -> та = функція перевантаження операції може бути оголошена лише як метод класу. Для інших операцій функції перевантаження операцій можуть не бути методами класу.

Оператори, які можна перевантажити:



Оператори, які не можна перевантажити:

- sizeof

- . (селектор елемента структури або класу)

- \* (оператор доступу до елементу за вказівником)

- :: (оператор дозволу видимості)

- ?: (тернарний оператор)

- typeid

- const\_cast

- dynamic\_cast

- reinterpret\_cast

- static\_cast

- # і ## (символи препроцесору)

***Завдання:***

Розширити функціональність розроблених у 4 лабораторній роботі класів за допомогою операторів, що задані варіантом та оператора присвоювання. Конкретні функції операторів реалізувати на власний розсуд (крім оператора присвоювання). Організувати виведення та введення даних за допомогою класів-потоків сin, cout та перевантажених операторів вводу/виводу. Написати програму, яка демонструє роботу з об'єктами цього класу.

******

*Код програми:*

#include <iostream>

#include <math.h>

#include<clocale>

#include <string>

using namespace std;

class cCar {

private:

string name;

int price, run, priceRun = 1.5;

double usedfuel = 0.0016;

public:

double fuel;

cCar()

{

}

cCar (double fuel)

{

this->fuel = fuel;

}

cCar(double fuel, int price, int run, string name) {

this->fuel = fuel;

this->price = price;

this->run = run;

this->name = name;

}

cCar(const cCar& other) {

this->fuel = other.fuel;

this->price = other.price;

this->run = other.run;

this->name = other.name;

}

virtual ~cCar() {

}

//operators

cCar& operator-- ()

{

cout << endl << "Operator --" << endl;

this->run--;

this->price--;

return \*this;

}

cCar& operator-= (cCar Car2)

{

fuel -= Car2.fuel;

return \*this;

}

cCar& operator+= (cCar Car2)

{

fuel += Car2.fuel;

return \*this;

}

bool operator ==(const cCar& other)

{

cout << endl << "Operator ==" << (this->fuel == other.fuel && this->price == other.price && this->run == other.run && this->name == other.name && this->priceRun == other.priceRun && this->usedfuel == other.usedfuel) << endl;

return this->fuel == other.fuel && this->price == other.price && this->run == other.run && this->name == other.name && this->priceRun == other.priceRun && this->usedfuel == other.usedfuel;

}

//function

void Movement(int pdistance) {

double use;

run = +(pdistance / 1000);

price -= (run \* priceRun);

use = usedfuel \* pdistance;

fuel = 70 - use;

}

void ShowData()

{

setlocale(LC\_CTYPE, "ukr");

cout << endl << "Назва машини: " << name;

cout << endl << "Кiлькiсть палива: " << fuel << " л";

cout << endl << "Розмiр пробiгу: " << run << "км";

cout << endl << "Вартiсть машини: " << price << " грн";

cout << endl << "Вартiсть пробiгу: " << priceRun << " грн/км";

}

//getters

int GET\_run()

{

return run;

}

int Get\_price()

{

return price;

}

int Get\_fuel()

{

return fuel;

}

};

cCar operator + (cCar Car, cCar Car2)

{

return cCar(Car.fuel + Car2.fuel);

}

int main()

{

string name = "Porsche 911 Coupe";

string name2 = "AUDI R8";

cCar Car(70.0 , 4058366 , 0, name);

cCar Car2(80.0, 3550000, 0, name2);

setlocale(LC\_CTYPE, "ukr");

while (true)

{

int choice;

cout << endl;

cout << endl;

cout << '\t' << " МЕНЮ:" << endl;

cout << '\t' << "Поточний стан машини-> press 1" << endl;

cout << '\t' << "Проїхатись ->press 2" << endl;

cout << '\t' << "Оператори ->press 3" << endl;

cout << '\t' << "Вийти з машини ->press 4" << endl;

cout << " Ваш вибiр:";

cin >> choice;

if (choice == 1)

{

Car.ShowData();

}

if (choice == 2)

{

int pdistance;

cout << "Введiть дистанцiю,яку хочете проЇхати(в метрах)" << endl;

cin >> pdistance;

Car.Movement(pdistance);

}

if (choice == 3)

{

cout << "Was:" << endl << "run: " << Car.GET\_run() << endl << "price: " << Car.Get\_price() << endl;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

--Car;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << endl << "Has become:" << endl << "run: " << Car.GET\_run() << endl << "price: " << Car.Get\_price() << endl;

cout << "================================================================" << endl;

cout << "Was before operator +=:" << endl << "fuel: " << Car.Get\_fuel() << endl;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

Car.fuel += Car2.fuel;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << endl << "Has become:" << endl << "fuel: " << Car.Get\_fuel() << endl;

cout << "================================================================" << endl;

cout << "Was before operator -=:" << endl << "fuel: " << Car.Get\_fuel() << endl;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

Car.fuel -= Car2.fuel;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << endl << "Has become:" << endl << "fuel: " << Car.Get\_fuel() << endl;

cout << "================================================================" << endl;

cout << "====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====";

cout << endl;

cout << endl << "Was before operator = :" << endl;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

Car == Car2;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "================================================================" << endl;

cout << endl;

cout << "Was:" << endl;

Car.ShowData();

cout << endl;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

Car = Car2;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << endl;

cout << "Has become:" << endl;

Car.ShowData();

cout << endl;

cout << "================================================================" << endl;

cout << endl;

cout << endl << "has become after operator = :" << endl;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

Car == Car2;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====";

}

if (choice == 4)

{

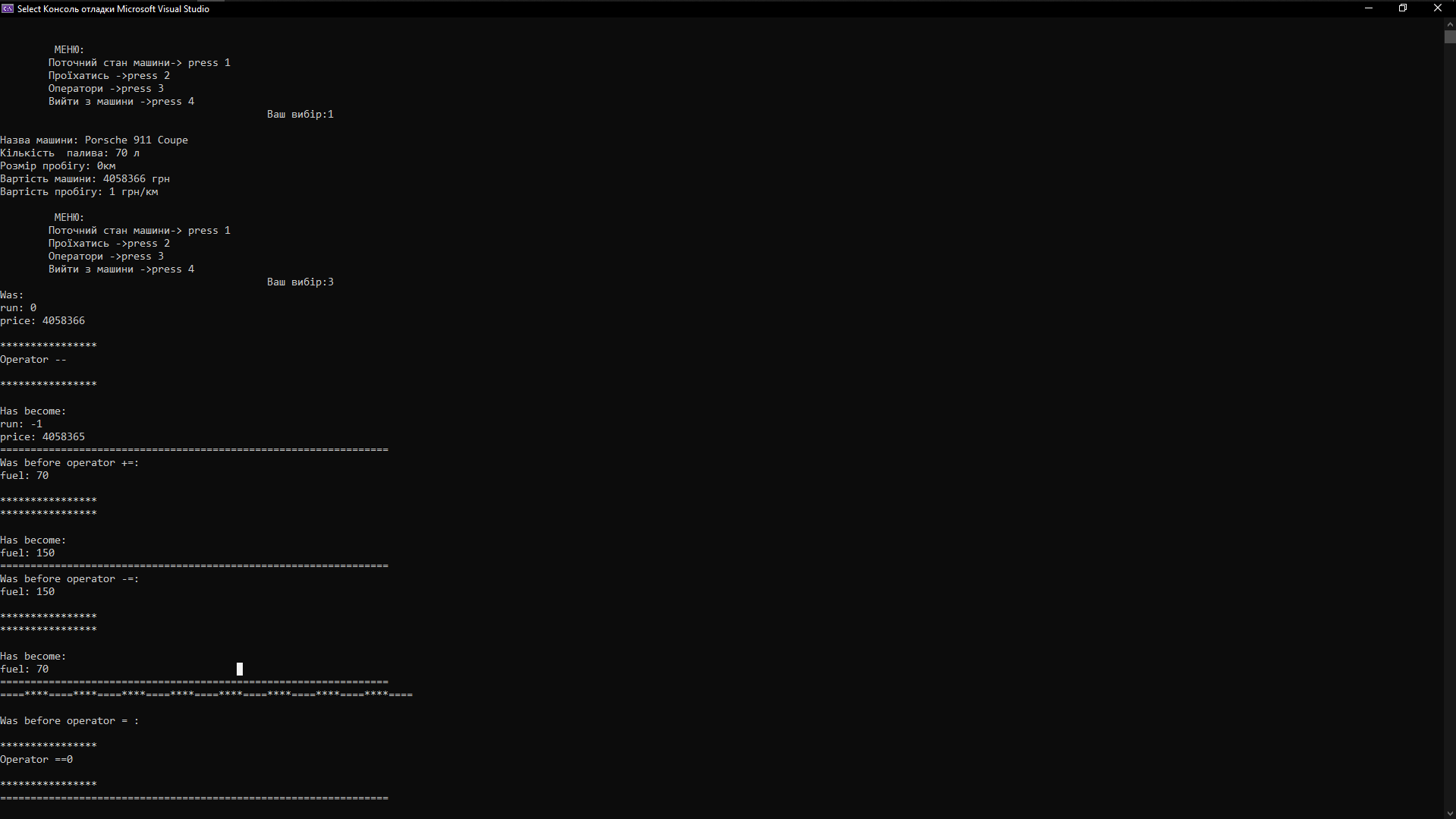
break;

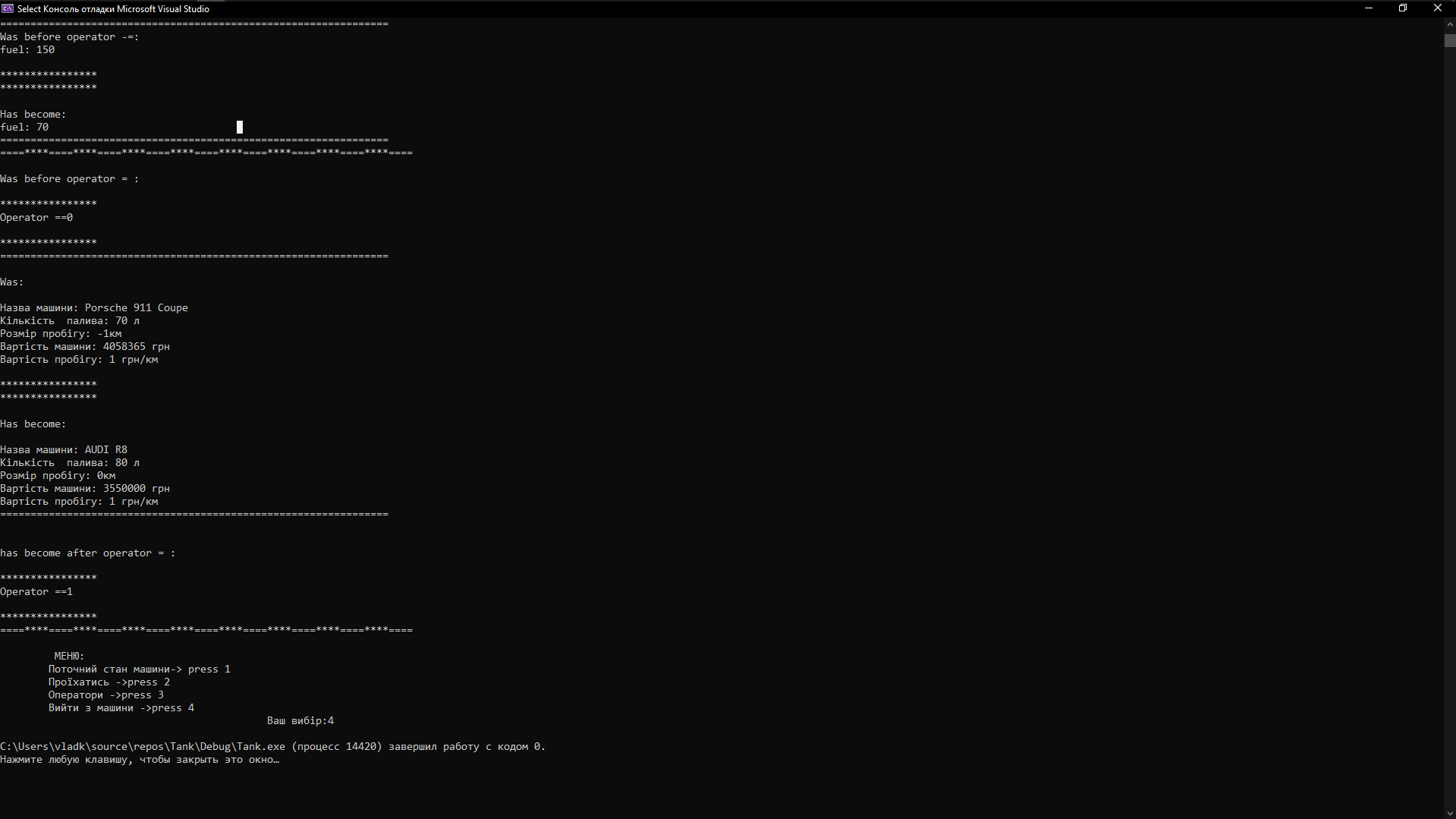
}

}

}

*Вікно результату:*

**

**

*Висновок:* я познайомився з таким «явищем», як перевантаження операторів.

Також пропрактикувався в їх написанні.