Практична робота №2

Тема: Робота із гілками Git-репозиторію. Перевантаження арифметичних операторів.

Мета: Закріпити знання про систему керування версіями Git та gitрепозиторій, навчитись працювати із гілками коду. Закріпити навички перевантаження арифметичних операторів для роботи із класами.

Хід роботи

1. Відкриваю у Git Bash створений у попередній практичній роботі репозиторій.

2. Створюю нову гілку коду із назвою PR2 і переключаюсь на цю гілку.

3. Для класу розробленого у попередній роботі визначаю оператор зчитування об’єкту з консольного потоку sdt::cin.

4. Перевіряю роботу перевантаженого оператора >> в функції main.

5. Роблю коміт зроблених змін коду проєкту із повідомлення «overload read object from std::cin».

6. Перевантажую оператор виводу об’єкта у потік sdt::cout для класу.

7. Перевіряю роботу перевантаженого оператора << в функції main.

8. Роблю коміт проєкту із вказанням повідомлення «overload writing object to std::cout».

9. Додаю до класу оператор порівняння «==», який буде перевіряє рівність усіх атрибутів. Перевіряю роботу перевантаженого оператора.

10. Роблю коміт проєкту з назвою «overload operator ==».

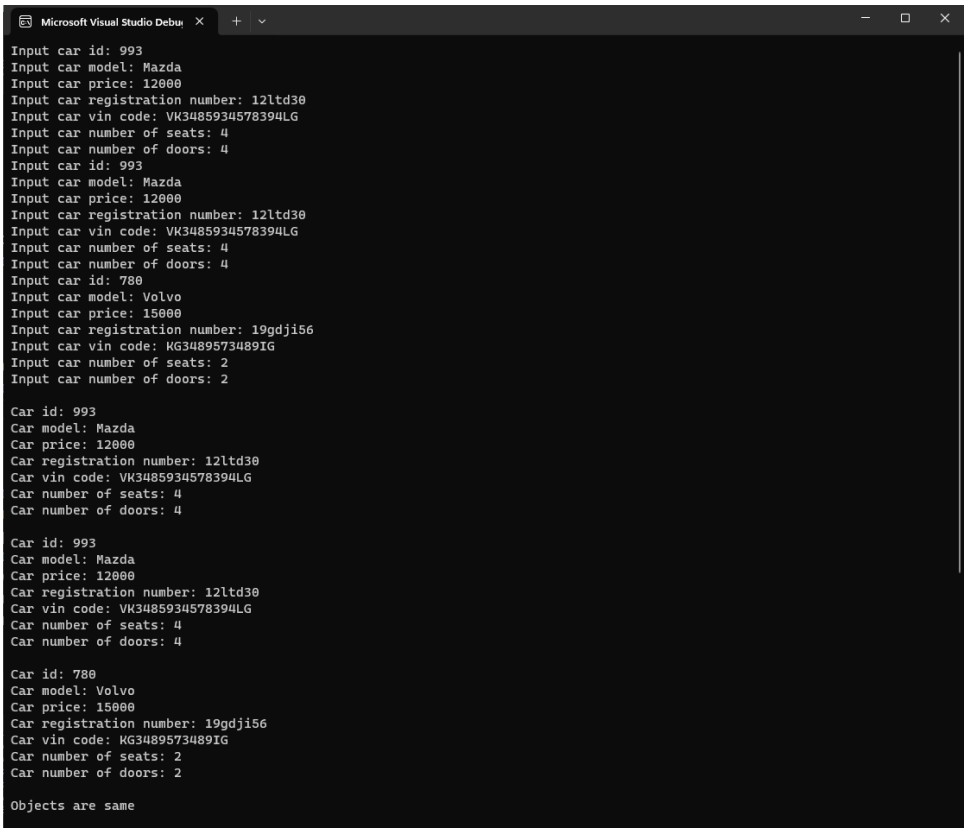
11. Переключаюсь у гілку main.

12. Об’єдную гілку PR2 з гілкою main.

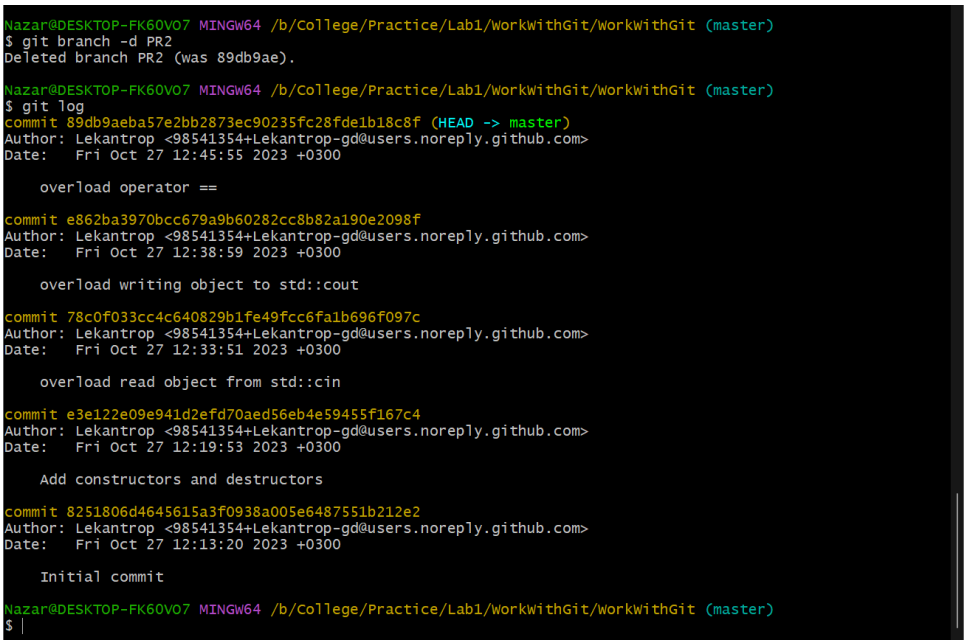
13. Перевіряю існування нових комітів з гілки PR2 у гілці main, після чого видаляю гілку PR2.

14. Оформлюю звіт.

Скріншот виконання програми:



Скріншот виконання команди git log:



Програмний код:

Файл main.cpp:

```
#include <iostream>
#include "Car.h"

int main()
{
```

```
    Car object1, object2, object3;
    std::cin >> object1 >> object2 >> object3;

    std::cout << endl << object1 << endl << object2 << endl << object3;

    cout << endl << (object1 == object2 ? "Objects are same" : "Objects are
different") << endl;
}
```

Файл Car.h:

```
#pragma once
#include<iostream>
using namespace std;

class Car
{
private:
int id;
string model;
int price;
string registrationNumber;
string vinCode;
int numberOfSeats;
int numberOfDoors;

public:
Car() = default;
Car(int id, const string& model, int price, const string& registrationNumber,
const string& vinCode, int numberOfSeats, int numberOfDoors);
Car(const Car &car);
~Car();
void input();
void output();
friend istream& operator >> (istream& in, Car& car);
friend ostream& operator << (ostream& out, Car& car);
bool operator==(const Car& other) const;
};
```

Файл Car.cpp:

```
#include "Car.h"

Car::Car(
int id,
const string& model,
int price,
const string& registrationNumber,
const string& vinCode,
int numberOfSeats,
int numberOfDoors)
:
```

```
id(id),
model(model),
price(price),
registrationNumber(registrationNumber),
vinCode(vinCode),
numberOfSeats(numberOfSeats),
numberOfDoors(numberOfDoors)
{
}

Car::Car(const Car& car)
{
this->id = car.id;
this->model = car.model;
this->price = car.price;
this->registrationNumber = car.registrationNumber;
this->vinCode = car.vinCode;
this->numberOfSeats = car.numberOfSeats;
this->numberOfDoors = car.numberOfDoors;
}

Car::~Car()
{
}

void Car::input()
{
cout << "Input car id: "; cin >> this->id;
cout << "Input car model: "; cin >> this->model;
cout << "Input car price: "; cin >> this->price;
cout << "Input car registration number: "; cin >> this->registrationNumber;
cout << "Input car vin code: "; cin >> this->vinCode;
cout << "Input car number of seats: "; cin >> this->numberOfSeats;
cout << "Input car number of doors: "; cin >> this->numberOfDoors;
}

void Car::output()
{
cout << "Car id: " << this->id << endl;
cout << "Car model: " << this->model << endl;
cout << "Car price: " << this->price << endl;
cout << "Car registration number: " << this->registrationNumber << endl;
cout << "Car vin code: " << this->vinCode << endl;
cout << "Car number of seats: " << this->numberOfSeats << endl;
cout << "Car number of doors: " << this->numberOfDoors << endl;
}

istream& operator>>(istream& in, Car& car)
{
car.input();
```

```
return in;
}

ostream& operator<<(ostream& out, Car& car)
{
car.output();
return out;
}

bool Car::operator==(const Car& other) const
{
return id == other.id &&
model == other.model &&
price == other.price &&
registrationNumber == other.registrationNumber &&
vinCode == other.vinCode &&
numberOfSeats == other.numberOfSeats &&
numberOfDoors == other.numberOfDoors;
}
```

Висновок: під час виконання даної практичної роботи, закріплено знання про систему керування версіями Git та git-репозиторій, вивчено принципи роботи із гілками коду. Закріплено навички перевантаження арифметичних операторів для роботи із класами.