Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ

Звіт

до лабораторної роботи № 5

з дисципліни: «Програмування, частина 2 (ООП)**»**

на тему: «Перевантаження операторів»

Варіант № 14

Підготував: Мишак М.А.

студент групи КІ-103

Перевірив:
Ст. викладач
Каф. ЕОМ
Гузинець Н.В

Львів 2025

Мета: познайомитися із перевантаженням операторів.

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 cin, cout та перевантажених операторів вводу/виводу. Написати програму, яка демонструє роботу з об'єктами цього класу.

14	+, --, !=, []

Теоретичний матеріал

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

Перевантаження операцій підпорядковується наступним правилам:

· При перевантаженні зберігаються кількість аргументів, пріоритети операцій та правила асоціації, що використовуються у стандартних типах даних;

· Для стандартних типів даних операції не підлягають перевизначенню;

· Перевантажена функція-оператор не може мати параметрів по замовчуванню, не успадковується та не може бути визначеною як static;

· Функція-оператор може бути визначена трьома способами – метод класу, дружня функція або звичайна функція. В останніх двох випадках вона повинна приймати хоча б один аргумент, що має тип класу, вказівника або посилання на клас.

Зауваження щодо перевантаження операцій:

· Неможливим є введення власних операторів.

· Компілятор C++ не розуміє семантики перевантаженого оператору, а отже, не нав'язує жодних математичних концепцій. Можна перевантажити, скажімо, оператор інкременту в якості зменшення аргументу, проте навряд чи в цьому є сенс.

· Не існує виведення складних операторів з простих: якщо ви перевантажили оператори operator+ та operator=, це зовсім не означає, що C++ обчислить вираз a += b, оскільки ви не перевантажили operator +=.

· Перевантаження бінарних операторів не тотожньо відносно перестановки аргументів місцями, тим більше, якщо вони різного типу.

Дружними до класу можуть бути: класи, методи та функції. Якщо якийсь з даних елементів має бути дружнім до класу, от в цьому класі його слід оголосити як дружній. Для цьо слід зробити:

- У випадку якщо дружнім має бути клас, то цей клас треба оголосити дружнім в тілі класу, до якого він має бути дружнім за допомогою ключового слова friend.

У цьому випадку всі елементи дружнього класу матимуть доступ до закритих і захищених елементів класу, до якого вони є дружніми. Оскільки тут someClass ідентифікується за допомогою ключового слова class як клас, то випереджуючого оголошення для компілятора робити не потрібно.

Лістинги (тексти**) програм:**

GasStation.h:

```
#include <iostream>
#include <string>
using namespace std;

class CGasStation {
private:
   double dieselVolume;
   double gasolineVolume;
   double lpgVolume;
   double electricVolume;
   double dieselReserve;
   double gasolineReserve;
   double lpgReserve;
   double electricReserve;
   double nozzleCapacity;

public:
   CGasStation(double dieselVolume = 1000, double gasolineVolume = 1000, double
lpgVolume = 1000, double electricVolume = 1000,
      double dieselReserve = 500, double gasolineReserve = 500, double lpgReserve =
500, double electricReserve = 500,
      double nozzleCapacity = 50);


   void get_info() const;

   void setDieselVolume(double volume);
   double getDieselVolume() const;
   void setGasolineVolume(double volume);
   double getGasolineVolume() const;
   void setLpgVolume(double volume);
   double getLpgVolume() const;
   void setElectricVolume(double volume);
   double getElectricVolume() const;
   void setDieselReserve(double reserve);
   double getDieselReserve() const;
   void setGasolineReserve(double reserve);
```

```
    double getGasolineReserve() const;
    void setLpgReserve(double reserve);
    double getLpgReserve() const;
    void setElectricReserve(double reserve);
    double getElectricReserve() const;
    void setNozzleCapacity(double capacity);
    double getNozzleCapacity() const;

    CGasStation& operator=(const CGasStation& other);
    CGasStation operator+(const CGasStation& other) const;
    CGasStation& operator--();
    bool operator!=(const CGasStation& other) const;
    double operator[](int index) const;
    friend ostream& operator<<(ostream& out, const CGasStation& station);
    friend istream& operator>>(istream& in, CGasStation& station);
};
```

GasStation.cpp:

```
#include "GasStation.h"

CGasStation::CGasStation(double dieselVolume, double gasolineVolume, double
lpgVolume, double electricVolume,
    double dieselReserve, double gasolineReserve, double lpgReserve, double
electricReserve,
    double nozzleCapacity)
    : dieselVolume(dieselVolume), gasolineVolume(gasolineVolume),
lpgVolume(lpgVolume), electricVolume(electricVolume),
    dieselReserve(dieselReserve), gasolineReserve(gasolineReserve),
lpgReserve(lpgReserve), electricReserve(electricReserve),
    nozzleCapacity(nozzleCapacity) {
}

void CGasStation::get_info() const {
    cout << *this;
}

void CGasStation::setDieselVolume(double volume) { dieselVolume = volume; }
double CGasStation::getDieselVolume() const { return dieselVolume; }
void CGasStation::setGasolineVolume(double volume) { gasolineVolume = volume; }
double CGasStation::getGasolineVolume() const { return gasolineVolume; }
void CGasStation::setLpgVolume(double volume) { lpgVolume = volume; }
double CGasStation::getLpgVolume() const { return lpgVolume; }
void CGasStation::setElectricVolume(double volume) { electricVolume = volume; }
double CGasStation::getElectricVolume() const { return electricVolume; }
```

```
void CGasStation::setDieselReserve(double reserve) { dieselReserve = reserve; }
double CGasStation::getDieselReserve() const { return dieselReserve; }
void CGasStation::setGasolineReserve(double reserve) { gasolineReserve = reserve; }
double CGasStation::getGasolineReserve() const { return gasolineReserve; }
void CGasStation::setLpgReserve(double reserve) { lpgReserve = reserve; }
double CGasStation::getLpgReserve() const { return lpgReserve; }
void CGasStation::setElectricReserve(double reserve) { electricReserve = reserve; }
double CGasStation::getElectricReserve() const { return electricReserve; }
void CGasStation::setNozzleCapacity(double capacity) { nozzleCapacity = capacity; }
double CGasStation::getNozzleCapacity() const { return nozzleCapacity; }

CGasStation CGasStation::operator+(const CGasStation& other) const {
    return CGasStation(dieselVolume + other.dieselVolume, gasolineVolume +
other.gasolineVolume,
        lpgVolume + other.lpgVolume, electricVolume + other.electricVolume,
        dieselReserve + other.dieselReserve, gasolineReserve + other.gasolineReserve,
        lpgReserve + other.lpgReserve, electricReserve + other.electricReserve,
        nozzleCapacity);
}

CGasStation& CGasStation::operator--() {
    dieselReserve -= 100;
    gasolineReserve -= 100;
    lpgReserve -= 100;
    electricReserve -= 100;
    return *this;
}

bool CGasStation::operator!=(const CGasStation& other) const {
    return dieselVolume != other.dieselVolume || gasolineVolume !=
other.gasolineVolume ||
        lpgVolume != other.lpgVolume || electricVolume != other.electricVolume ||
        dieselReserve != other.dieselReserve || gasolineReserve != other.gasolineReserve
||
        lpgReserve != other.lpgReserve || electricReserve != other.electricReserve;
}
CGasStation& CGasStation::operator=(const CGasStation& other) {
    if (this != &other) {
        dieselVolume = other.dieselVolume;
        gasolineVolume = other.gasolineVolume;
        lpgVolume = other.lpgVolume;
        electricVolume = other.electricVolume;
        dieselReserve = other.dieselReserve;
```

```
        gasolineReserve = other.gasolineReserve;
        lpgReserve = other.lpgReserve;
        electricReserve = other.electricReserve;
        nozzleCapacity = other.nozzleCapacity;
    }
    return *this;
}


double CGasStation::operator[](int index) const {
    switch (index) {
    case 0:
        return dieselReserve;
    case 1:
        return gasolineReserve;
    case 2:
        return lpgReserve;
    case 3:
        return electricReserve;
    default:
        throw out_of_range("Недійсний індекс");
    }
}

ostream& operator<<(ostream& out, const CGasStation& station) {
    out << "Об'єм дизельного палива: " << station.dieselVolume << " л\n"
        << "Об'єм бензину: " << station.gasolineVolume << " л\n"
        << "Об'єм LPG: " << station.lpgVolume << " л\n"
        << "Об'єм електроенергії: " << station.electricVolume << " кВт-год\n"
        << "Запас дизельного палива: " << station.dieselReserve << " л\n"
        << "Запас бензину: " << station.gasolineReserve << " л\n"
        << "Запас LPG: " << station.lpgReserve << " л\n"
        << "Запас електроенергії: " << station.electricReserve << " кВт-год\n"
        << "Пропускна здатність заправочного пістолета: " << station.nozzleCapacity
<< " л/сек\n";
    return out;
}

istream& operator>>(istream& in, CGasStation& station) {
    cout << "Введіть об'єм дизельного палива: ";
    in >> station.dieselVolume;
    cout << "Введіть об'єм бензину: ";
    in >> station.gasolineVolume;
```

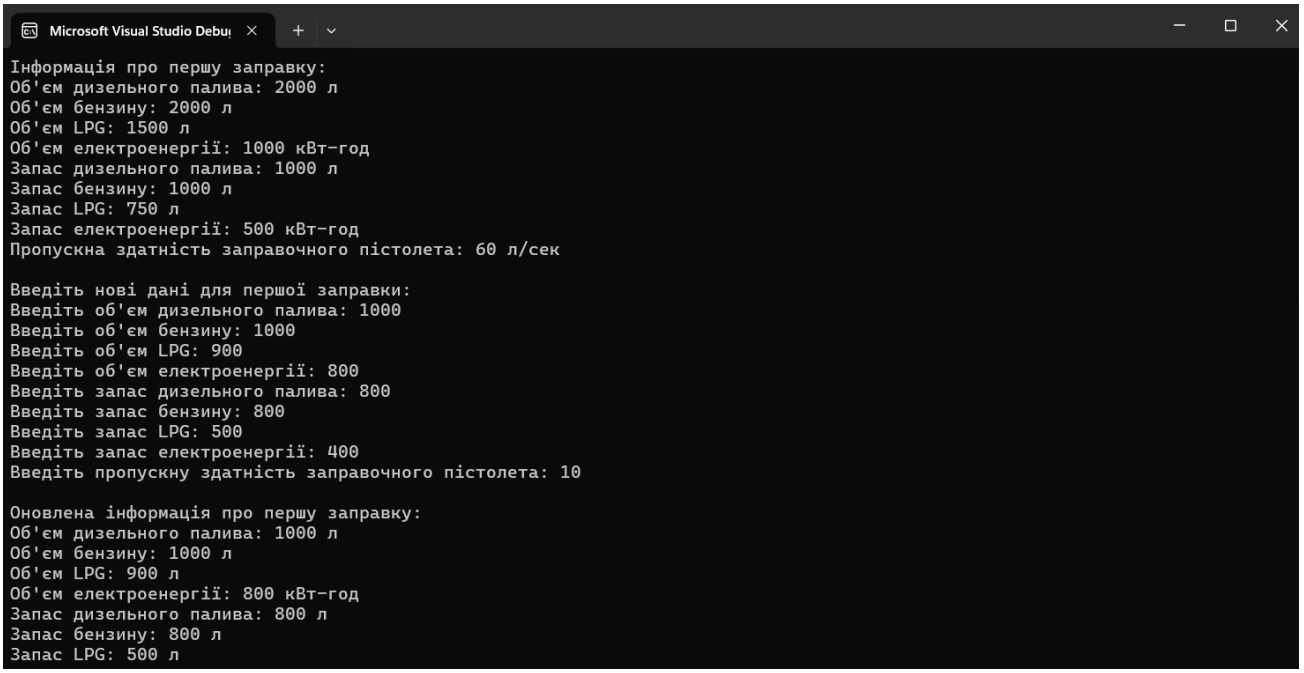
```
    cout << "Введіть об'єм LPG: ";
    in >> station.lpgVolume;
    cout << "Введіть об'єм електроенергії: ";
    in >> station.electricVolume;
    cout << "Введіть запас дизельного палива: ";
    in >> station.dieselReserve;
    cout << "Введіть запас бензину: ";
    in >> station.gasolineReserve;
    cout << "Введіть запас LPG: ";
    in >> station.lpgReserve;
    cout << "Введіть запас електроенергії: ";
    in >> station.electricReserve;
    cout << "Введіть пропускну здатність заправочного пістолета: ";
    in >> station.nozzleCapacity;
    return in;
}
```

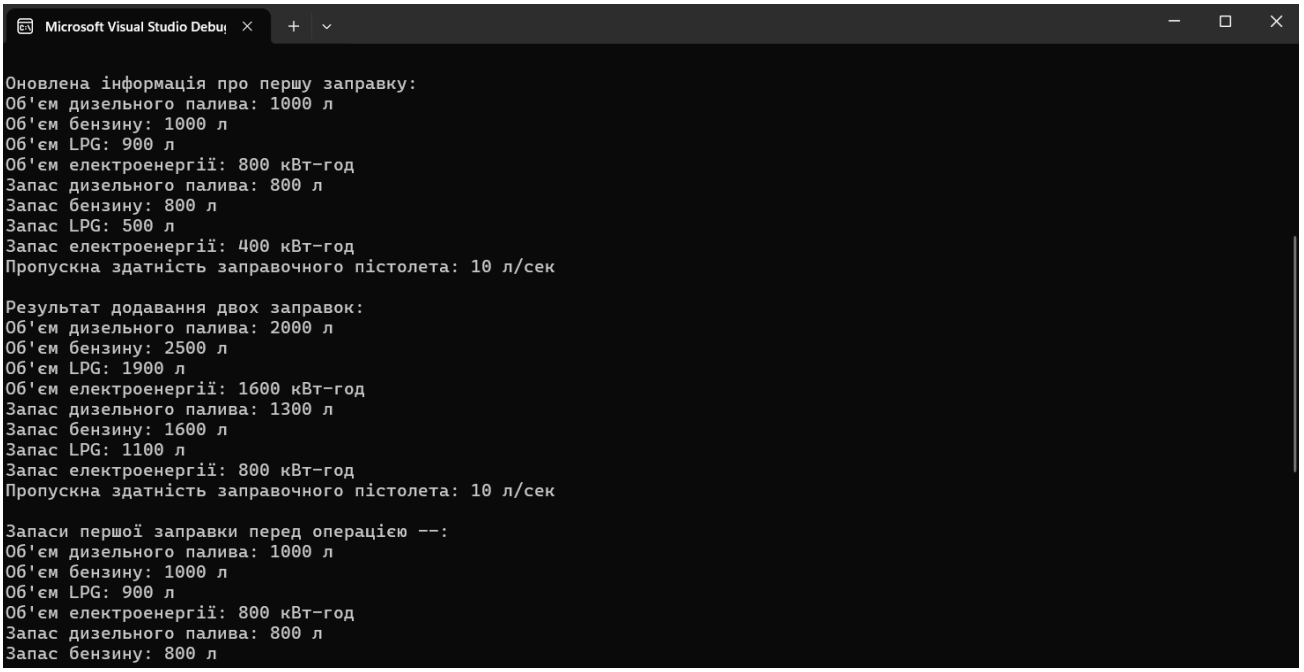
main.cpp:

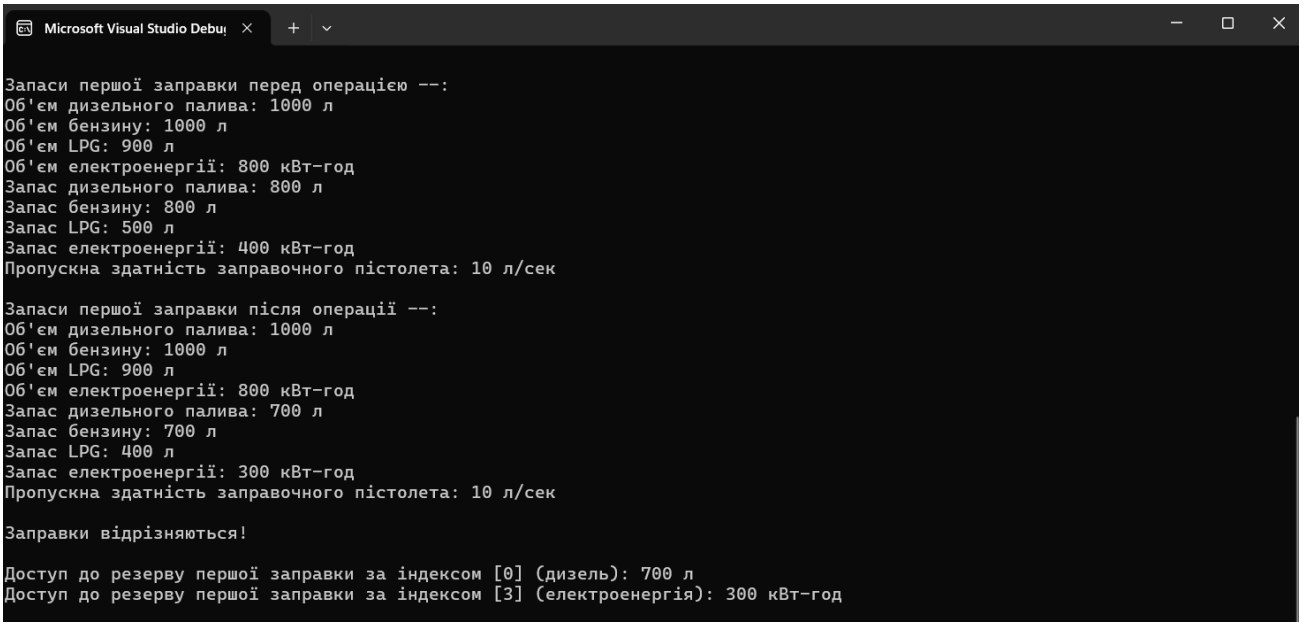
```
#include <iostream>
#include <string>
#include "GasStation.h"

// +, --, !=, []

using namespace std;

int main() {
    setlocale(LC_ALL, "Ukrainian");


    CGasStation station1(2000, 2000, 1500, 1000, 1000, 1000, 750, 500, 60);
    CGasStation station2(1000, 1500, 1000, 800, 500, 800, 600, 400, 50);


    cout << "Інформація про першу заправку:" << endl;
    cout << station1;


    cout << "\nВведіть нові дані для першої заправки:" << endl;
    cin >> station1;

    cout << "\nОновлена інформація про першу заправку:" << endl;
    cout << station1;
```

```
    CGasStation combinedStation = station1 + station2;
    cout << "\nРезультат додавання двох заправок:" << endl;
    cout << combinedStation;


    cout << "\nЗапаси першої заправки перед операцією --:" << endl;
    cout << station1;
    --station1;
    cout << "\nЗапаси першої заправки після операції --:" << endl;
    cout << station1;


    if (station1 != station2) {
        cout << "\nЗаправки відрізняються!" << endl;
    }
    else {
        cout << "\nЗаправки однакові!" << endl;
    }


    try {
        cout << "\nДоступ до резерву першої заправки за індексом [0] (дизель): "
            << station1[0] << " л" << endl;
        cout << "Доступ до резерву першої заправки за індексом [3]
(електроенергія): "
            << station1[3] << " кВт-год" << endl;
    }
    catch (const out_of_range& e) {
        cout << e.what() << endl;
    }

    return 0;
}
```

Результати виконання програм:







Висновок: На даній лабораторній роботі я познайомився з перевантаженням у мові програмування C++. На основі цих знань розширив функціональність розроблених у 4 лабораторній роботі класів за допомогою операторів, що задані варіантом та оператора присвоювання.