Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



**Звіт**

З лабораторної роботи № 5

З дисципліни «Об’єктно-орієнтоване програмування»

**На тему**: «Перевантаження операторів»

**Виконав:** ст. гр. КІ-15

Колосенко Д.О.

**Прийняв:** Козак Н.Б.

Львів 2020

**Мета:** познайомитися із перевантаженням операторів.

**Теоретичні відомості**

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

*тип operator символОператору (списокПараметрів)*

*{*

*//тіло методу*

*}*

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.

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.

**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.

|  |  |
| --- | --- |
| 14 | +,--,!=,[] |

**Код програми:**

**Header**

#pragma once

#include <iostream>

#include <iomanip>

using namespace std;

class CGasStation

{

private:

struct TypeGas

{

string Gas\_\_Name;

int Amount, MaxAmount;

};

int MaxLitres, Amout\_\_Litres;

double Capacity;

string GasNames[4] = { "Diesel","Gasoline","Methanol","Propane" };

TypeGas Dgases[4];

void Initialize()

{

for (int i = 0; i < 4; i++)

{

Dgases[i].Gas\_\_Name = GasNames[i];

Dgases[i].Amount = Amout\_\_Litres;

Dgases[i].MaxAmount = MaxLitres;

}

}

public:

friend ostream& operator<< (ostream& out, const CGasStation& gass);

friend bool operator==(const CGasStation& gass1, const CGasStation& gass2);

CGasStation()

{

MaxLitres = 1500;

Amout\_\_Litres = 750;

Capacity = 0.5;

Initialize();

}

CGasStation(int MaxLitres, int Amout\_\_Litres, double Capacity)

{

this->MaxLitres = MaxLitres;

this->Amout\_\_Litres = Amout\_\_Litres;

this->Capacity = Capacity;

Initialize();

}

void Information()

{

cout << setw(15) << "Gas name" << setw(15) << "Now litres" << setw(15) << "Max Litres" << endl;

for (int i = 0; i < 4; i++)

{

cout << setw(15) << Dgases[i].Gas\_\_Name << setw(15) << Dgases[i].Amount << setw(15) << Dgases[i].MaxAmount << endl;

}

}

void AddAmount(int ZAmount, int Where)

{

if (ZAmount+Dgases[Where].Amount< Dgases[Where].MaxAmount)

{

Dgases[Where].Amount += ZAmount;

}

else

{

cout << "Maximum";

Dgases[Where].Amount = Dgases[Where].MaxAmount;

}

}

void Refuil(int Litres, int Where)

{

double time;

if (Dgases[Where].Amount- Litres >0)

{

time = Litres / Capacity;

Dgases[Where].Amount -= Litres;

cout << "It will took: " << time << " secunds"<<endl;

}

else

{

cout << "Not enough gas" << endl;

}

}

CGasStation operator+(CGasStation gass1)

{

gass1.Amout\_\_Litres += Amout\_\_Litres;

gass1.Capacity += Capacity;

gass1.MaxLitres += MaxLitres;

for (int i = 0; i < 4; i++)

{

gass1.Dgases[i].Amount += Dgases[i].Amount;

gass1.Dgases[i].MaxAmount += Dgases[i].MaxAmount;

}

return gass1;

}

TypeGas& operator[](int index)

{

if (index >= 4 && index < 0)

{

cout << "Array index out of bound, exiting";

exit(0);

}

return Dgases[index];

}

CGasStation& operator--(int)

{

MaxLitres--;

Amout\_\_Litres--;

Capacity--;

for (int i = 0; i < 4; i++)

{

Dgases[i].Amount--;

Dgases[i].MaxAmount--;

}

return \*this;

}

};

ostream& operator<< (ostream& out, const CGasStation& gass)

{

out << setw(15) << "Gas name" << setw(15) << "Now litres" << setw(15) << "Max Litres" << endl;

for (int i = 0; i < 4; i++)

{

out << setw(15) << gass.Dgases[i].Gas\_\_Name << setw(15) << gass.Dgases[i].Amount << setw(15) << gass.Dgases[i].MaxAmount << endl;

}

return out;

}

bool operator==(const CGasStation& gass1, const CGasStation& gass2)

{

bool check=true;

if (gass1.MaxLitres == gass2.MaxLitres && gass1.Capacity == gass2.Capacity && gass1.Amout\_\_Litres == gass2.Amout\_\_Litres) check = true;

else

{

return false;

}

for (int i = 0; i < 4; i++)

{

if (gass1.Dgases[i].Amount == gass2.Dgases[i].Amount && gass1.Dgases[i].Gas\_\_Name == gass2.Dgases[i].Gas\_\_Name && gass1.Dgases[i].MaxAmount == gass2.Dgases[i].MaxAmount)check = true;

else

{

return false;

}

}

return true;

}

bool operator!=(const CGasStation& gass1, const CGasStation& gass2)

{

return !(gass1 == gass2);

}

**Main**

#include <iostream>

#include "Gas.h"

using namespace std;

int main()

{

CGasStation GasStation;

cout << GasStation<<endl;

GasStation--;

cout << GasStation << endl;

CGasStation GasStation2;

if (GasStation != GasStation2)

{

cout << "Not equal 1 and 2\n";

}

if (GasStation == GasStation)

{

cout << "Equal 1 and 1\n\n";

}

CGasStation GasStation3 = GasStation + GasStation2;

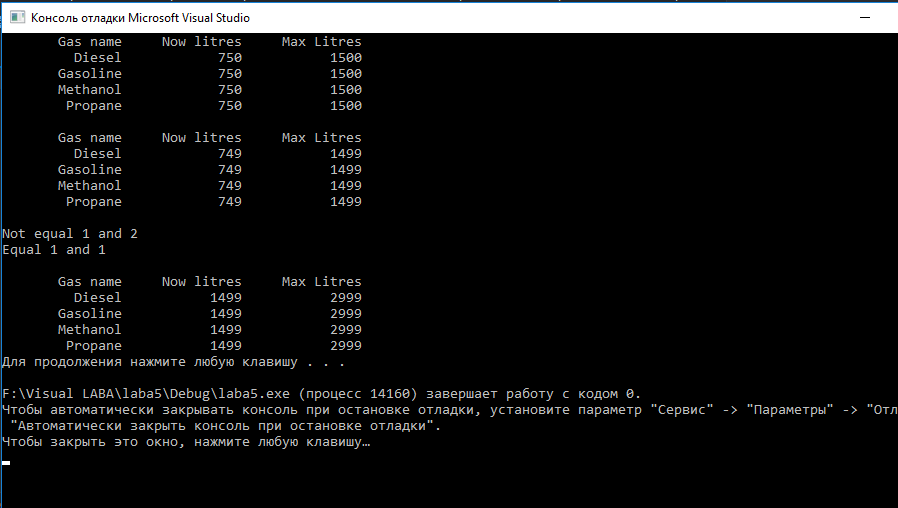
cout << GasStation3;

system("pause");

return 0;

}

**Результат програми:**



**Висновок:** Під час виконання цієї лабораторної роботи я познайомився із перевантаженням операторів.