МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



Звіт до лабораторних робіт

з ООП

Виконав:

ст. групи КІ-15

Ковальський П.А.

Прийняв:

викл. Козак Н.Б.

Львів-2020

ЛАБОРАТОРНА РОБОТА № 5.

ПЕРЕВАНТАЖЕННЯ ОПЕРАТОРІВ

**Мета:** 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: 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. 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 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)

**Варіант-12**

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 <string>

using namespace std;

class CBrowser

{

public:

CBrowser();

CBrowser(std::string adress, double height, double width, double inSpeed, double outSpeed)

{

set\_PageAdress(adress);

set\_PageHeight(height);

set\_PageWidth(width);

set\_downloadSpeed(inSpeed);

set\_outloadSpeed(outSpeed);

}

~CBrowser();

void set\_PageAdress(std::string adress)

{

currentPageAdress = adress;

}

void set\_PageHeight(double height)

{

PageHeight = height;

}

void set\_PageWidth(double width)

{

PageWidth = width;

}

void set\_downloadSpeed(double inSpeed)

{

downloadSpeed = inSpeed;

}

void set\_outloadSpeed(double outSpeed)

{

outloadSpeed = outSpeed;

}

std::string get\_PageAdress()

{

return currentPageAdress;

}

double get\_PageHeight()

{

return PageHeight;

}

double get\_PageWidth()

{

return PageWidth;

}

double get\_downloadSpeed()

{

return downloadSpeed;

}

double get\_outloadSpeed()

{

return outloadSpeed;

}

double GetSizeOfSpeed(double sizeofFile)

{

double time = 0.0;

double speed;

speed = get\_downloadSpeed();

time = sizeofFile / speed;

return time;

}

double Download(double sizeofFile)

{

return GetSizeOfSpeed(sizeofFile);

}

double Outload(double sizeofFile)

{

double time = 0;

double speed;

speed = get\_outloadSpeed();

time = sizeofFile / speed;

return time;

}

void Show\_status()

{

cout << "Page adress: " << get\_PageAdress() << endl;

cout << "Page height: " << get\_PageHeight() << endl;

cout << "Page width: " << get\_PageWidth() << endl;

cout << "Page download speed: " << get\_downloadSpeed() << endl;

cout << "Page outload speed: " << get\_outloadSpeed() << endl;

}

ostream& operator<< (ostream& output)

{

output << this->currentPageAdress << endl;

output << this->PageHeight << endl;

output << this->PageWidth << endl;

output << this->downloadSpeed << endl;

output << this->outloadSpeed << endl;

return output;

}

CBrowser& operator= (CBrowser& b)

{

this->currentPageAdress = b.currentPageAdress;

this->PageHeight = b.PageHeight;

this->PageWidth = b.PageWidth;

this->downloadSpeed = b.downloadSpeed;

this->outloadSpeed = b.outloadSpeed;

return \*this;

}

CBrowser& operator+= (CBrowser& b)

{

this->downloadSpeed = this->downloadSpeed + b.downloadSpeed;

this->outloadSpeed = this->outloadSpeed + b.outloadSpeed;

return \*this;

}

CBrowser& operator-= (CBrowser& b)

{

this->downloadSpeed = this->downloadSpeed - b.downloadSpeed;

this->outloadSpeed = this->outloadSpeed - b.outloadSpeed;

return \*this;

}

const bool operator>(CBrowser& b)

{

if (this->downloadSpeed + this->outloadSpeed > b.downloadSpeed + b.outloadSpeed)

{

return 1;

}

return 0;

}

const bool operator<(CBrowser& b)

{

if (this->downloadSpeed + this->outloadSpeed < b.downloadSpeed + b.outloadSpeed)

{

return 1;

}

return 0;

}

private:

string currentPageAdress;

double PageHeight;

double PageWidth;

double downloadSpeed;

double outloadSpeed;

};

void Start()

{

CBrowser \*browser1;

CBrowser \*browser2;

double time[2];

string pagePath;

double sizeY;

double sizeX;

double speed\_in;

double speed\_out;

std::string pagePath2;

double sizeY2;

double sizeX2;

double speed\_in2;

double speed\_out2;

cout << "Page 1 adress: ";

getline(cin, pagePath);

cout << "Page 1 height: ";

cin >> sizeX;

cout << "Page 1 width: ";

cin >> sizeY;

cout << "Page 1 download speed: ";

cin >> speed\_in;

cout << "Page 1 outload speed: ";

cin >> speed\_out;

browser1 = new CBrowser(pagePath, sizeX, sizeY, speed\_in, speed\_out);

getchar();

cout << endl;

cout << "Page 2 adress: ";

getline(cin, pagePath2);

cout << "Page 2 height: ";

cin >> sizeX2;

cout << "Page 2 width: ";

cin >> sizeY2;

cout << "Page 2 download speed: ";

cin >> speed\_in2;

cout << "Page 2 outload speed: ";

cin >> speed\_out2;

browser2 = new CBrowser(pagePath2, sizeX2, sizeY2, speed\_in2, speed\_out2);

cout << endl;

time[0] = browser1->Download(500.8);

cout << "Page 1 time to download 500.8 Mb file : " << time[0] << endl;

time[1] = browser1->Download(345.6);

cout << "Page 2 time to download 345.6 Mb file : " << time[1] << endl;

cout << endl;

time[0] = browser1->Outload(500.8);

cout << "Page 1 time to outload 500.8 Mb file : " << time[0] << endl;

time[1] = browser1->Outload(345.6);

cout << "Page 2 time to outload 345.6 Mb file : " << time[1] << endl;

cout << endl;

cout << "Page 1 status : " << endl;

browser1->Show\_status();

cout << endl;

cout << "Page 2 status : " << endl;

browser2->Show\_status();

cout << "\nOverload += : " << endl;

\*browser1 += \*browser2;

browser1->Show\_status();

cout << "\nOverload -= : " << endl;

\*browser1 -= \*browser2;

browser1->Show\_status();

bool a;

cout << "\nOverload > : " << endl;

a = browser1 > browser2;

cout << "browser1 speed > browser2 speed? = " << a << endl;

cout << "\nOverload < : " << endl;

a = browser1 < browser2;

cout << "browser1 speed < browser2 speed? = " << a << endl;

cout << "\nOverload = : " << endl;

browser1 = browser2;

browser1->Show\_status();

}

int main()

{

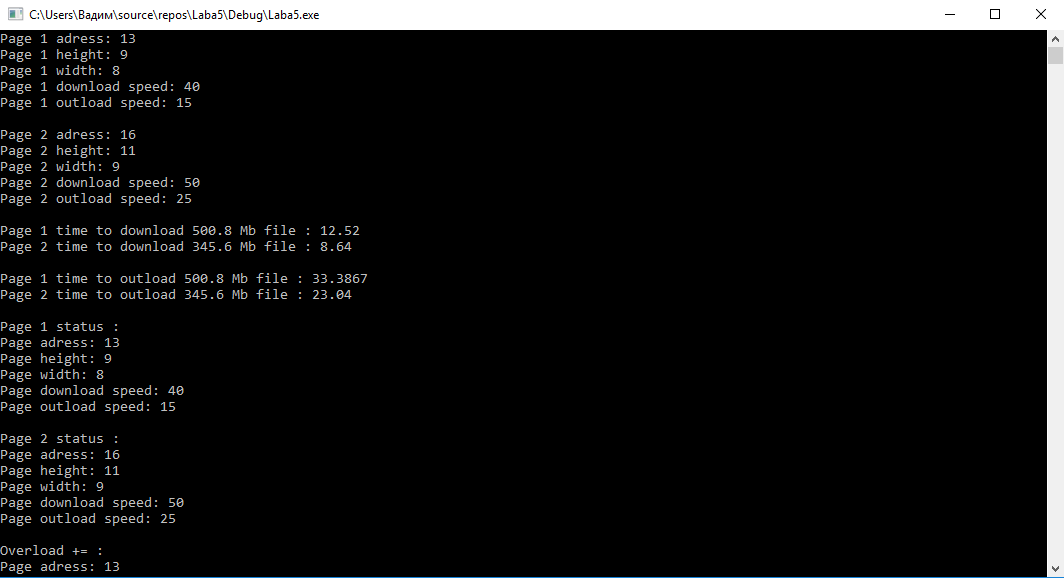
Start();

system("pause");

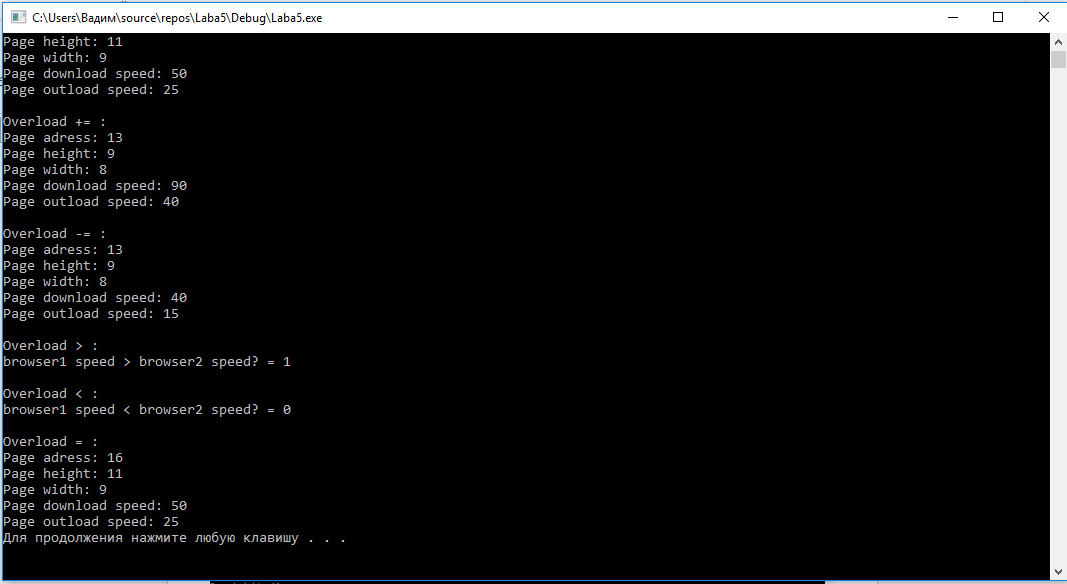
return 0;

}

**Результат:**

**1)**

**2)**



**Висновок:**

На даній лабораторній роботі я ознайомився із перевантаженням операторів.