Міністерство освіти і науки

Національний університет “Львівська політехніка”

**Кафедра ЕОМ**



**Звіт**

з лабораторної роботи № 5

# з дисципліни: “ Програмування, частина 2 (Об'єктно-орієнтоване програмування”

на тему: “ ПЕРЕВАНТАЖЕННЯ ОПЕРАТОРІВ ”

Виконав: ст. гр. КІ-15 Гвоздь Ю. А.

Прийняв: Козак Н. Б.

Львів – 2020

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. 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) Коли операція реалізована як метод класу, тоді: - якщо операція є унарною (передбачає один операнд, наприклад, інкременту або []), то лівим операндом вважаєтсья об‘єкт, до якого застосовується операція, а правий операнд відсутній, тому метод, що реалізує даний оператор не приймає аргументів, за виключенням кількох операторів (наприклад, постфіксна форма інкременту або декременту). - якщо операція є бінарною (передбачає 2 операнди, наприклад додавання або присвоєння) і лівий операнд є об‘єктом класу у якому перевантажується операція, то крайній лівий операнд вважається об‘єктом зпід якого здійснюється виклик даного методу (оператора), а правий операнд – передається як параметр, тому для нього слід вказати тип аргумента у методі; перевантажених операторів одного виду може бути кілька в залежності від типу аргументу, що передається методу; Якщо операція є бінарною але лівий операнд не є об‘єктом класу у якому перевантажується операція, то дана операція не може бути методом класу, а має бути реалізована як дружня функція, якщо ця функція повинна мати прямий доступ до закритих або захищених елементів цього класу, або звичайна функція в протилежному випадку. Перевантажена операція << повинна мати лівий операнд типу ostream & (такий, як cout), так що вона не може бути функцією-елементом. Аналогічно, перевантажена операція >> повинна мати лівий операнд типу іstream & (такий, як cіn), так що вона теж не може бути функцією-елементом. До того ж кожна з цих перевантажених функцій-операцій може забажати доступу до закритих елементів-даних об'єкта класу, так що ці перевантажені функції-операції роблять функціями-друзями класу. Будь-яку бінарну операцію можна перевантажувати як нестатичний метод з одним аргументом, або як функцію, що не є елементом, із двома аргументами (один з цих аргументів повинен бути або об'єктом класу, або посиланням на об'єкт класу). Унарну операцію класу можна перевантажувати як метод без аргументів, або як функцію, з одним аргументом; цей аргумент повинен бути або об'єктом класу, або посиланням на об'єкт класу. Функції-елементи, що реалізують перевантажені операції, повинні бути нестатичними, щоб вони могли мати доступ до даних класу. Нагадаємо, що статичні методи можуть мати доступ тільки до статичних даних-елементів класу. При перевантаженні унарних операцій переважно створюють методи класу, замість дружніх функцій, що не є членами класу. Дружніх функцій краще уникати доти, поки вони не стануть абсолютно необхідними. Використання друзів порушує інкапсуляцію класу. 80 Щоб перевантажити операцію інкремента та декремента для одержання можливості використання і префіксної, і постфіксної форм, кожна з цих двох перевантажених функцій-операцій повинна мати різну сигнатуру, щоб компілятор мав можливість визначити, яка версія мається на увазі в кожному конкретному випадку. Префіксний варіант перевантажується як будь-яка інша префіксна унарна операція. Для постфіксної форми вводиться додатковий параметр цілого типу у список аргументів, щоб зробити функцію для постфіксного варіанту відмінною від функції для префіксної форми. Зауваження щодо перевантаження операцій: • Неможливим є введення власних операторів. • Компілятор С++ 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 С++ обчислить вираз a += b, оскільки ви не перевантажили operator +=. • Перевантаження бінарних операторів не тотожньо відносно перестановки аргументів місцями, тим більше, якщо вони різного типу. Синтаксис виклику операторів реалізований у мові С++ таким чином, щоб програміст міг записати операцію над об‘єктом класу з звичному для нього вигляді, наприклад object1 += object2. Проте для компілятора такий запис стосовно об‘єктів не є звичним, бо дана операція у такому вигляді визначена лише для простих типів, а для об‘єкту класу даний виклик має бути перетворений у виклик методу, що оголошений в класі, або у виклик відповідної функції. Тому при компіляції компілятор перетворює даний виклик у виклик відповідного методу або функції за наступним принципом: - якщо оператор реалізований у вигляді методу, то лівий операнд перетворюється в об‘єкт з-під якого викликається метод, що реалізує вказаний оператор, а правий операнд, якщо він присутній, передається методу в якості аргументу. - якщо оператор реалізований у вигляді функції або дружньої функції, то воклик операції перетворюєтсья у виклик функції, яка приймає своїм лівим параметром лівий операнд операції, а правим, якщо такий присутній, - правий операнд операції.

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.



Виконання завдання:

Код програми:

#include <iostream>

#include <string>

#include <ctime>

#include <fstream>

using namespace std;

class CCoins

{

private:

const char\* sface = "Лицева сторона";

const char\* sback = "Тильна сторона";

int coins[3] = { 0,0,0 };

int\* asd = coins;

double quantity = 0;

double all = 0;

double perc = 0;

public:

CCoins operator ++ (int)

{

CCoins ret(\*this);

srand((int)time(0));

for (int i = 0; i < 3; i++) {

all++;

coins[i] = rand() % 2 + 1;

//cout << coins[i] << endl;

if (coins[i] == 1) {

quantity++;

} //Якщо coins[i]==1, тоді це лицева сторона,інакше тильна.

}

return ret;

}

bool operator == (const CCoins& other)

{

int f = 0;

for (int i = 0; i < 3; i++) {

if (this->coins[i] == other.coins[i]) {

f++;

}

}

if (f == 3)

return true;

else

return false;

}

CCoins operator () ()

{

CCoins ret(\*this);

for (int i = 0; i < 3; i++) {

if (coins[i] == 1)

cout << sface << endl;

else

cout << sback << endl;

}

cout << "Кiлькiсть монет що є лицевими: " << quantity << endl;

return ret;

}

CCoins operator % (int)

{

CCoins ret(\*this);

perc = 100 / all \* quantity;

cout << perc << "%" << endl;

return ret;

}

int operator = (int)

{

quantity = 0;

all = 0;

for (int i = 0; i < 3; i++) {

coins[i] = 0;

}

return 0;

}

friend ostream& operator << (ostream& i, const CCoins val)

{

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

cout << "Пiдкинути монети.(натиснiть 1)" << endl;

cout << "Поточний стан?.(натиснiть 2)" << endl;

cout << "Скинути кiлькiсть лицевих монет.(натиснiть 3)" << endl;

cout << "Кинути монети знову i перевiрити, чи збiгаються лицевi сторони?(натисiнiть 4)" << endl;

cout << "Вiдсоток монет з лицевими сторонами.(натиснiть 5)" << endl;

cout << "Перевернути монету.(натиснiть 6)" << endl;

cout << "Зупинити програму.(натиснiть 7)" << endl;

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

return i;

}

friend istream& operator>>(istream& i, CCoins& st)

{

int s;

cout << "Яку монету ви хочете перевернути? (Натиснiть 1 або 2 або 3)" << endl;

cin >> s;

cout << "Якщо ви хочете перевернути на лицеву сторону (натиснiть 1)" << endl;

cout << "Якщо ви хочете перевернути на тильну сторону (натиснiть 2)" << endl;

i >> st.coins[s - 1];

return i;

}

protected:

};

int main()

{

setlocale(LC\_ALL, "ukr");

int func = 1;

CCoins num1;

CCoins num2;

cout << "Що вам потрiбно?" << endl;

while (func != 7) {

cout << num1;

cin >> func;

if (func == 1) {

num1++;

}

if (func == 2) {

num1();

}

if (func == 3) {

num1 = 0;

if (func == 1) {

num1++;

}

if (func == 2) {

num1();

}

}

if (func == 4) {

num1();

num2++;

num2();

bool result = num1 == num2;

cout << result << endl;

}

if (func == 5) {

num1 % 1;

}

if (func == 6) {

cin >> num1;

}

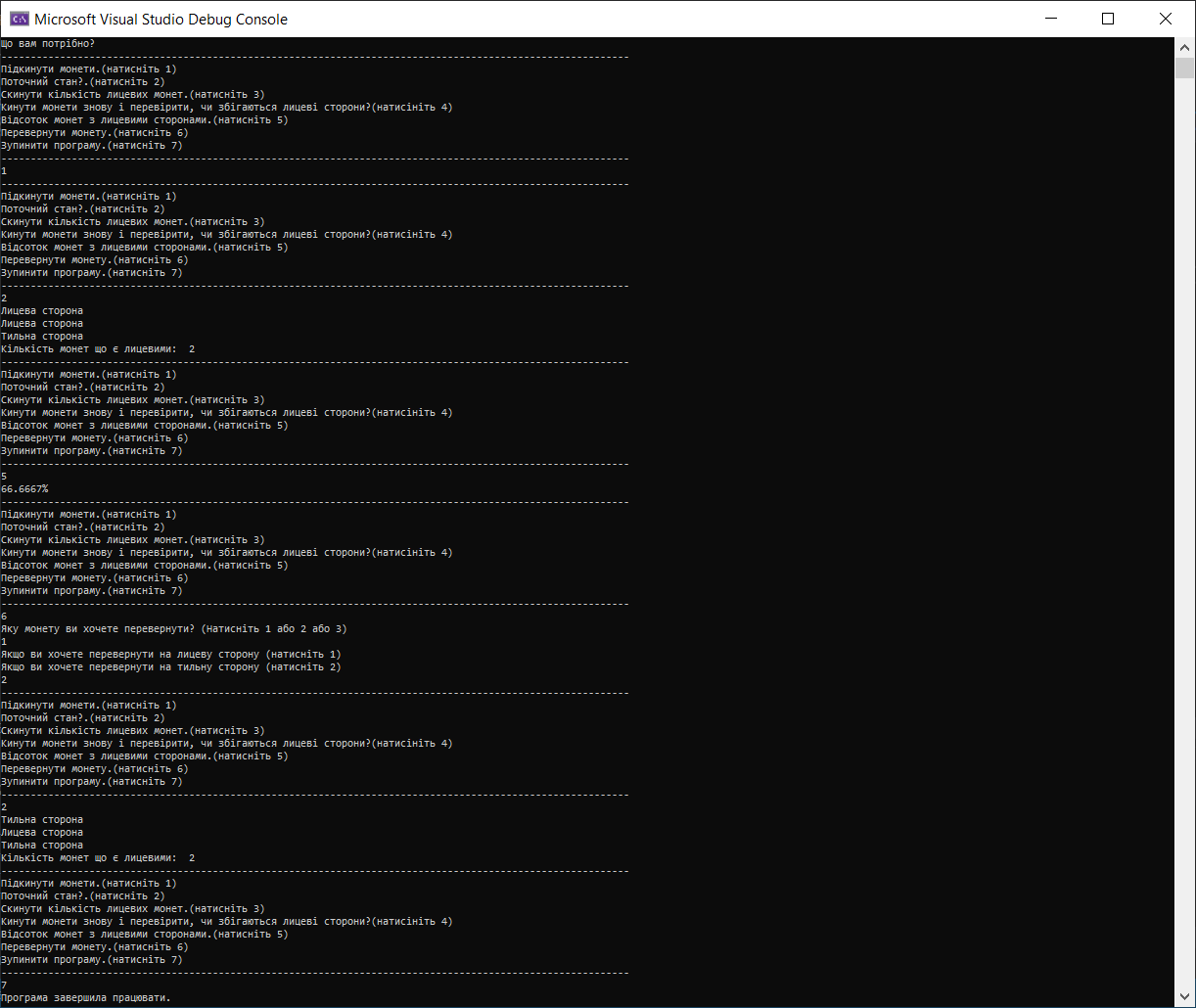
}

cout << "Програма завершила працювати.";

return 0;

}

Результат роботи програми:



Висновок: на цій лабораторній роботі я ознайомився із перевантаженням операторів.