**Міністерство освіти і науки України**

**Національний університет „Львівська політехніка”**

**Кафедра ЕОМ**

****

**Звіт**

**з лабораторної роботи №5**

**з дисципліни: “Програмування, ч.2 (ООП)”**

**на тему: “ Перевантаження операторів ”**

**Варіант 22**

**Виконав: ст.гр. КІ-15**

**Петрущак Я.В.**

**Прийняв: викладач**

**Козак Н.Б.**

**Львів**

**2020**

**Теоретичні відомості**

Мета: познайомитися із перевантаженням операторів.

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

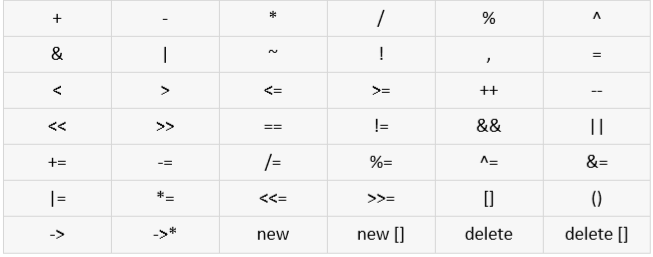
Щоб використовувати операцію над об'єктами класів, ця операція повинна бути перевантажена, але є два виключення. Операції присвоювання (=) і взяття адреси (&) створюються в класі автоматично за замовчуванням, тому їх можна використовувати без явного перевантаження. За замовчуванням операція присвоювання 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 можна перевантажити:



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.

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.

**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.



**Варіант 22 – код програми.**

#include <iostream>

#define size 30

class CMachinegun {

private:

uint16\_t capacity,

targetsHit;

double hitProb;

long long resource;

public:

CMachinegun& operator += (uint16\_t x) {

this->capacity += x;

return \*this;

}

CMachinegun& operator -= (uint16\_t x) {

this->capacity -= x;

return \*this;

}

friend bool operator == (const CMachinegun& obj, const CMachinegun& obj1) {

if (obj.capacity == obj1.capacity)

return true;

else return false;

}

friend bool operator > (const CMachinegun& obj, const CMachinegun& obj1) {

if (obj.capacity > obj1.capacity)

return true;

else return false;

}

CMachinegun(uint16\_t x, uint16\_t y, double z, long long n) {

this->capacity = x;

this->targetsHit = y;

this->hitProb = z;

this->resource = n;

}

void setCapacity(uint16\_t x) {

this->capacity = x;

}

void setTargetsHit(uint16\_t x) {

this->targetsHit = x;

}

void setHitProb(double x) {

this->hitProb = x;

}

uint16\_t getCapacity() {

return this->capacity;

}

uint16\_t getTargetsHit() {

return this->targetsHit;

}

double getHitProb() {

return this->hitProb;

}

void reloadGun() {

std::cout << "Gun has been reloaded!\n";

this->capacity = size;

}

void unloadGun() {

std::cout << "Gun has been unloaded!\n";

this->capacity = 0;

}

void shoot() {

if (capacity && resource) {

if (rand() % 100 + 0 > this->hitProb \* 100) {

std::cout << "Target has been hit!\n"; targetsHit++;

}

else std::cout << "Target hasn`t been hit!\n";

capacity--;

resource--;

}

else std::cout << "No ammo!\n";

}

void show() {

std::cout << "Capacity: " << this->capacity << "\nTargets hit: " << this->targetsHit <<

"\nHit probability: " << this->hitProb << "\nResource left: " << this->resource << "\n";

}

};

uint16\_t main() {

CMachinegun obj(30, 0, 0.45, 1000);

CMachinegun obj1(30, 0, 0.45, 1000);

std::cout << obj.getCapacity() << "\n";

std::cout << obj.getHitProb() << "\n";

std::cout << obj.getTargetsHit() << "\n";

obj.shoot();

obj.unloadGun();

obj.reloadGun();

obj.shoot();

std::cout << obj.getTargetsHit() << "\n";

obj += 20;

std::cout << obj.getCapacity() << "\n";

obj -= 20;

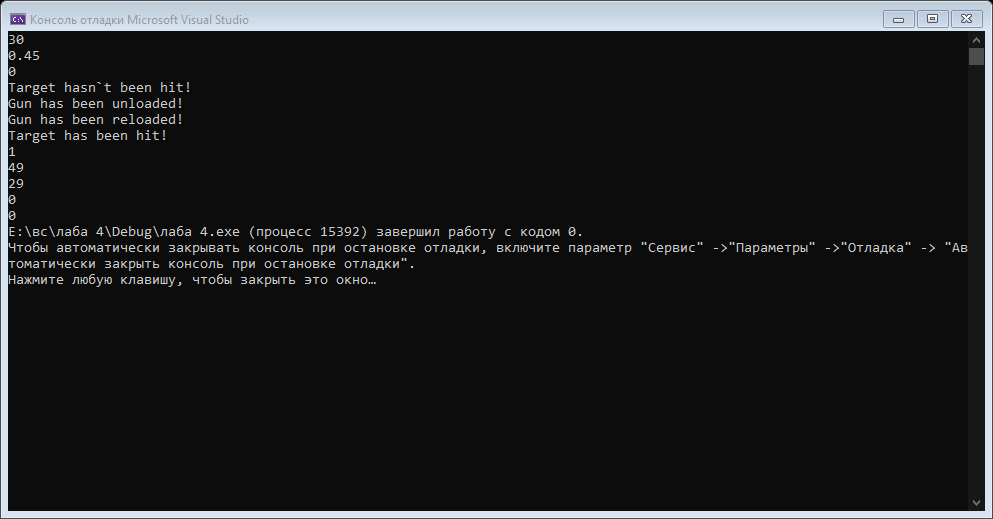
std::cout << obj.getCapacity() << "\n";

std::cout << (obj == obj1) << "\n";

std::cout << (obj > obj1);

return 0;

}



**Висновок:** під час виконання лабораторної роботи я ознайомився з властивостями первантаження функцій .