Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра “Електронних обчислювальних машин”



**Звіт з лабораторної роботи №5**

**Програмування, частина 2**

**(Об’єктно - орієнтоване програмування)**

на тему:

“Перевантаження операторів”

Варіант - 25

**Виконала:**

ст. гр. КІ-15

Романішина А. Ю.

**Перевірив:**

асист. каф. ЕОМ

Козак Н. Б.

Львів - 2020

**Мета:** познайомитися із перевантаженням операторів.

**Теоретичні відомості:**

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 з- під 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.

**Індивідуальне 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.



**Код програми:**

#include <iostream>

#include <iomanip>

#include <Windows.h>

#include <vector>

#include <string>

**using** **namespace** std;

**class** CFlat

{

**private**:

vector <string> HumansName;

**int** time,

Square,

electricity,

water,

gas;

**public**:

CFlat(**int** set\_time, **int** set\_Square, **int** set\_electricity, **int** set\_water, **int** set\_gas)

{

setState(set\_time, set\_Square, set\_electricity, set\_water, set\_gas);

}

**void** addHuman()

{

cout << "Введіть ім'я" << endl;

string name;

cin >> name;

HumansName.push\_back(name);

}

**void** viewHuman()

{

**for** (**unsigned** **int** i = 0; i < HumansName.size(); i++)

{

cout << i << "\t" << HumansName[i] << endl;

}

}

**int** getCountPeople()

{

**return** HumansName.size();

}

**void** delHuman()

{

viewHuman();

cout << "Введіть номер" << endl;

**int** k;

cin >> k;

HumansName.erase(HumansName.begin() + k);

}

**void** setState(**int** time, **int** Square, **int** electricity, **int** water, **int** gas)

{

**this**->time = time;

**this**->Square = Square;

**this**->electricity = electricity;

**this**->water = water;

**this**->gas = gas;

}

**void** currentstate()

{

cout << "Термін експлуатації квартири: " << time << " дн" << endl;

cout << "Площа: " << Square << "м3" << endl;

cout << "Кількість спожитої електроенергії: " << electricity << " Вт/г" << endl;

cout << "Кількість спожитої води: " << water << " л" << endl;

cout << "Кількість спожитого газу: " << gas << " кб" << endl;

system("pause");

}

**double** price()

{

**return** electricity \* 1.2 + water \* 2.1 + gas \* 1.5;

}

**friend** ostream& **operator** << (ostream& os, **const** CFlat& Flat) {

os << "Термін експлуатації квартири: " << Flat.time << " дн" << endl;

os << "Площа: " << Flat.Square << "м3" << endl;

os << "Кількість спожитої електроенергії: " << Flat.electricity << " Вт/г" << endl;

os << "Кількість спожитої води: " << Flat.water << " л" << endl;

os << "Кількість спожитого газу: " << Flat.gas << " кб" << endl;

os << endl;

**return** os;

}

**friend** istream& **operator** >> (istream& is, CFlat& Flat) {

cout << "Введіть термін експлуатації квартири: " << endl;

is >> Flat.time;

cout << "Введіть площу: " << endl;

cin >> Flat.Square;

cout << "Введіть кількість спожитої електроенергії: " << endl;

is >> Flat.electricity;

cout << "Введіть кількість спожитої води: " << endl;

is >> Flat.water;

cout << "Введіть кількість спожитого газу: " << endl;

is >> Flat.gas;

**return** is;

}

CFlat& **operator** = (**const** CFlat& Flat)

{

**if** (&Flat == **this**)

{

**return** \***this**;

}

**else**

{

setState(Flat.time, Flat.Square, Flat.electricity, Flat.water, Flat.gas);

HumansName = Flat.HumansName;

**return** \***this**;

}

}

CFlat& **operator** += (**const** CFlat& Flat)

{

time += Flat.time;

Square += Flat.Square;

electricity += Flat.electricity;

water += Flat.water;

gas += Flat.gas;

**for** (**unsigned** **int** i = 0; i < Flat.HumansName.size(); i++)

{

HumansName.push\_back(Flat.HumansName[i]);

}

**return** \***this**;

}

CFlat **operator** -- (**int**)

{

electricity--;

water--;

gas--;

**return** \***this**;

}

**bool** **operator** != (**const** CFlat& Flat)

{

**if** (time == Flat.time &&

Square == Flat.Square &&

electricity == Flat.electricity &&

water == Flat.water &&

gas == Flat.gas &&

HumansName.size() == Flat.HumansName.size()

)

{

**return** 0;

}

**else**

**return** 1;

}

};

**extern** **void** mainScreen(CFlat& Flat);

CFlat N1(50, 15, 7, 50, 80), N2(0, 0, 0, 0, 0);

**void** FlatSelector()

{

system("cls");

cout << "Виберіть дію" << endl;

cout << "1. Перейти до керування першою квартирою" << endl;

cout << "2. Перейти до керування другою квартирою" << endl;

cout << "3. Скопіювати дані про першу квартиру в другу" << endl;

cout << "4. Додати дані з першої квартири в другу" << endl;

cout << "5. Зменшити параметри першої квартири на 1" << endl;

cout << "6. Зменшити параметри другої квартири на 1" << endl;

cout << "7. Перевірити чи квартири не однакові" << endl;

cout << "8. Вихід" << endl;

**int** ch;

cin >> ch;

**switch** (ch)

{

**case** 1:

system("cls");

mainScreen(N1);

**case** 2:

system("cls");

mainScreen(N2);

**case** 3:

system("cls");

N2 = N1;

FlatSelector();

**case** 4:

system("cls");

N2 += N1;

FlatSelector();

**case** 5:

system("cls");

N1--;

FlatSelector();

**case** 6:

system("cls");

N2--;

FlatSelector();

**case** 7:

system("cls");

**if** (N1 != N2)

{

cout << "Квартири не однакові" << endl;

}

**else**

{

cout << "Квартири однакові" << endl;

}

system("pause");

FlatSelector();

**default**:

exit(0);

}

}

**extern** **void** mainScreen(CFlat& Flat)

{

system("cls");

cout << "Виберіть дію" << endl;

cout << "1. Заселити мешканця" << endl;

cout << "2. Переглянути імена мешканців" << endl;

cout << "3. Переглянути кількість мешканців" << endl;

cout << "4. Виселити мешканця" << endl;

cout << "5. Переглянути поточний стан" << endl;

cout << "6. Змінити стан" << endl;

cout << "7. Розрахувати вартість" << endl;

cout << "8. Повернутись до попереднього меню" << endl;

**int** ch;

cin >> ch;

**switch** (ch)

{

**case** 1:

system("cls");

Flat.addHuman();

mainScreen(Flat);

**case** 2:

system("cls");

Flat.viewHuman();

system("pause");

mainScreen(Flat);

**case** 3:

system("cls");

cout << "Кількість мешканців: " << Flat.getCountPeople() << endl;

system("pause");

mainScreen(Flat);

**case** 4:

system("cls");

Flat.delHuman();

mainScreen(Flat);

**case** 5:

system("cls");

Flat.currentstate();

mainScreen(Flat);

**case** 6:

system("cls");

cin >> Flat;

mainScreen(Flat);

**case** 7:

system("cls");

cout << "Вартість спожитих ресурсів: " << Flat.price() << "грн" << endl;

system("pause");

mainScreen(Flat);

**default**:

FlatSelector();

}

}

**int** main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

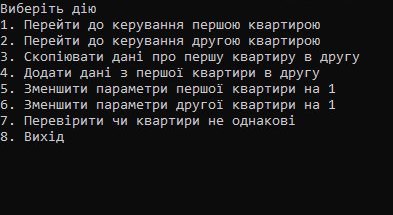
FlatSelector();

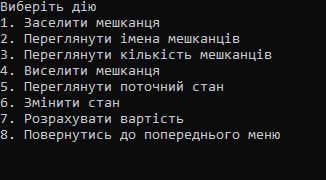
system("pause");

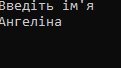
**return** 0;

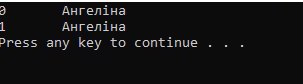
}

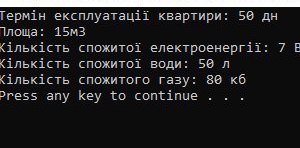
**Результат виконання функції:**

****

****

****

****

****

**Висновок**: на даній лабораторній роботі я познайомилась з перевантаженням операторів.