Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

Дисципліна: Об’єктно-орієнтоване програмування

**Лабораторна робота №5**

**Тема:** **«**Перевантаження операторів

|  |
| --- |
| Виконав: ст. гр. КБ-23 |
| Захарченко А.О. |
| Перевірив: викладач  Козірова Наталія Леонідівна |
|  |

Кропивницький

2024

**Тема:** Перевантаження операторів

**Мета:** ознайомитись з поняттям перевантаження операторів та навчитись їх програмно реалізовувати мовою С++.

**Завдання 1:** Перевантажте оператори для завдання № 2 з лабораторної роботи №2.

**Реалізація**

**Person.cpp:**

#include "Person.h"

using namespace std;

void Person::setname(const string name) {

this->name = name;

}

void Person::setage(int age) {

this->age = age;

}

void Person::setaddress(const string address) {

this->address = address;

}

string Person::getname() {

return name;

}

int Person::getage() {

return age;

}

string Person::getaddress() {

return address;

}

Person::Person() : name("no\_data"), age(0), address("no\_data") { //Конструктор за замовчуванням

cout << "Usual constr " << endl; //Теж

}

Person::Person(const string name, int age, string address) : name(name), age(age), address(address) {

cout << "Paramentr constr" << endl;

}

Person::Person(const Person& other) : name(other.name), age(other.age), address(other.address) {

cout << "Copy constr" << endl;

}

;

Person& Person::operator=(const Person& other) {

if (this != &other)

{

name = other.name;

age = other.age;

address = other.address;

}

return \*this;

}

bool Person::operator==(const Person& other) const

{

return (name == other.name && age == other.age && address == other.address);

}

ostream& operator<<(ostream& os, const Person& person)

{

os << "Name: " << person.name << " Age: " << person.age << " Address: " << person.address;

return os;

}

/\*Person::~Person() {

cout << "Destructor for this one: " << endl;

}

\*/

**Person.h:**

#pragma once

#ifndef PERSON\_H

#define PERSON\_H

#include <string>

#include <iostream>

using namespace std;

class Person

{

private:

string name;

int age;

string address;

public:

Person();

Person(string name, int age, string address);

Person(const Person& other);

string getname();

int getage();

string getaddress();

void setname(string name);

void setage(int age);

void setaddress(string address);

Person& operator=(const Person& other);

bool operator==(const Person& other)const;

friend ostream& operator<<(ostream& os, const Person& person);

//~Person();

};

#endif

**main.cpp:**

#include <iostream>

#include "Person.h"

using namespace std;

int main() {

Person person;

string name;

int age;

string address;

cout << "Name: ";

cin >> name;

person.setname(name);

cout << "Age: ";

cin >> age;

person.setage(age);

cout << "Address: ";

cin >> address;

person.setaddress("address");

cout << "Name: " << person.getname() << endl;

cout << "Age: " << person.getage() << endl;

cout << "Address: " << person.getaddress() << endl;

Person person1("Josh", 35, "Luvabur");

{

cout << "Name: " << person1.getname() << endl;

cout << "Age: " << person1.getage() << endl;

cout << "Address: " << person1.getaddress() << endl;

}

/\*Person person3 = person1;

cout << "Name: " << person3.getname() << endl;

cout << "Age: " << person3.getage() << endl;

cout << "Address: " << person3.getaddress() << endl;

\*/

person = person;

if (person1 == person)

{

cout << "Same human " << endl;

}

else

{

cout << "Not the same human " << endl;

}

cout << person1 << endl;

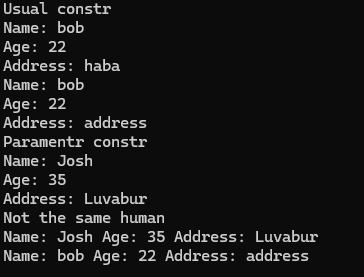
cout << person << endl;

return 0;

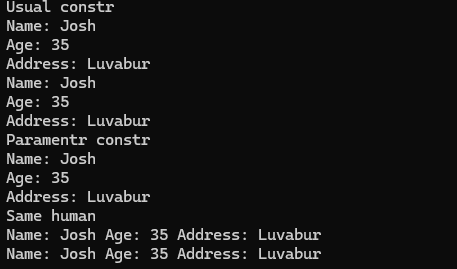
}

**Результат:**

В першому випадку користувач задає різні дані, і як результат ми маємо такий результат.

****

У результаті 2 при введені користувачем однакових даних отримуємо результат Same human

****

**Примітка:** під час виконання даного завдання необхідно вимкнути оператор присвоєння.

**Завдання 2:** Продовжить розробку гри «Спіймай муху», тепер ваша муха має стати класом, реалізуйте декілька рівнів гри, на кожному наступному рівні, має з’являтися нова муха, яка буде класом наслідником. Також, з другого рівня на вашій формі має з’явитися вікно, муха має намагатися вилетіти в це вікно, задача гравця відігнати муху від вікна і загнати її в пастку.

**Реалізація**

**Результат:**

**Завдання 3(Варіант 3):** Створіть клас Matrix, який представляє квадратну матрицю розміром NxN. В класі Matrix перевантажте наступні оператори:

1. Оператор + для додавання двох матриць.
2. Оператор - для віднімання двох матриць.
3. Оператор \* для множення двох матриць.
4. Оператор \* для множення матриці на скаляр.
5. Оператор == для порівняння двох матриць на рівність.
6. Оператор != для порівняння двох матриць на нерівність.
7. Оператор << для виводу матриці на екран.

Додайте в клас також необхідні конструктори, деструктор та інші методи, які можуть знадобитись для роботи з матрицями.

Напишіть програму, де ви використовуєте цей клас та перевірте роботу всіх перевантажених операторів. Створіть декілька об'єктів класу Matrix і виконайте з ними операції додавання, віднімання, множення, порівняння на рівність та виведення на екран.

**Реалізація**

**Matrix.cpp**

#include "Matrix.h"

#include <iostream>

using namespace std;

Matrix::Matrix(int n) : size(n),data(n,vector<int>(n,0)) {}

Matrix::Matrix(const Matrix& other) : size(other.size), data(other.data) {}

Matrix::~Matrix() {}

Matrix Matrix::operator+(const Matrix& other) const {

Matrix result(size);

for (int a = 0; a < size; ++a)

for (int b = 0; b < size; ++b)

result.data[a][b] = data[a][b] + other.data[a][b];

return result;

}

Matrix Matrix::operator-(const Matrix& other) const {

Matrix result(size);

for (int a = 0; a < size; ++a)

for (int b = 0; b < size; ++b)

result.data[a][b] = data[a][b] - other.data[a][b];

return result;

}

Matrix Matrix::operator\*(const Matrix& other) const {

Matrix result(size);

for (int a = 0; a < size; ++a)

for (int b = 0; b < size; ++b)

for (int k = 0; k < size; ++k)

result.data[a][b] += data[a][k] \* other.data[k][b];

return result;

}

Matrix Matrix::operator\*(int scalar) const {

Matrix result(size);

for (int a = 0; a < size; ++a)

for (int b = 0; b < size; ++b)

result.data[a][b] = data[a][b] + scalar;

return result;

}

bool Matrix::operator==(const Matrix& other) const {

return data == other.data;

}

bool Matrix::operator!=(const Matrix& other) const {

return !(\*this == other);

}

ostream& operator<<(ostream& os, const Matrix& matrix) {

for (const auto& row : matrix.data) {

for (int val : row)

os << val << " ";

os << "\n";

}

return os;

}

void Matrix::setValue(int a, int b, int value) {

data[a][b] = value;

}

int Matrix::getValue(int a, int b) const {

return data[a][b];

}

void Matrix::inputMatrix() {

cout << "Enter your number: " << size << "x" << size << "\n";

for (int a = 0; a < size;++a) {

for (int b = 0;b < size;++b) {

cout << "Element [" << a << "][" << b << "] ";

cin >> data[a][b];

}

}

}

**Matrix.h**

#pragma once

#include <iostream>

#include <vector>

using namespace std;

class Matrix

{

private:

vector<vector<int>> data;

int size;

public:

Matrix(int n);

Matrix(const Matrix& other);

~Matrix();

Matrix operator+(const Matrix& other) const;

Matrix operator-(const Matrix& other) const;

Matrix operator\*(const Matrix& other) const;

Matrix operator\*(int scalar) const;

bool operator==(const Matrix& other) const;

bool operator!=(const Matrix& other) const;

friend ostream& operator<<(ostream & os, const Matrix& matrix);

void setValue(int a, int b, int value);

int getValue(int a, int b) const;

void inputMatrix();

};

**Source.cpp**

#include <iostream>

#include "Matrix.h"

using namespace std;

int main() {

int n;

cout << "Enter size your Matrix: ";

cin >> n;

Matrix mat1(n), mat2(n);

cout << "Enter value for the first: \n";

mat1.inputMatrix();

cout << "Enter value for the second: \n";

mat2.inputMatrix();

Matrix sum = mat1 + mat2;

Matrix min = mat1 - mat2;

Matrix prod = mat1 \* mat2;

Matrix scalarprod = mat1 \* 2;

cout << "Mat 1: \n" << mat1;

cout << "Mat 2: \n" << mat2;

cout << "Mat sum is: \n" << sum;

cout << "Mat min is: \n" << min;

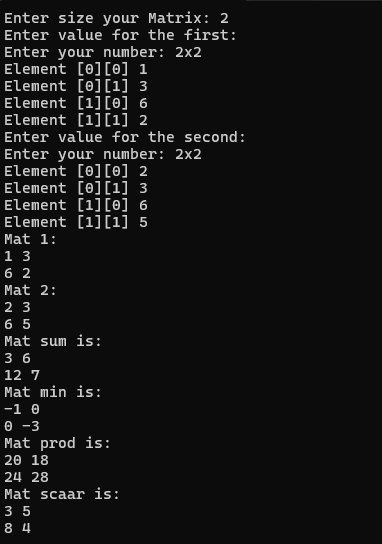
cout << "Mat prod is: \n" << prod;

cout << "Mat scaar is: \n" << scalarprod;

return 0;

}

**Результат:**

****

**Висновок:**

Виконавши роботу було опрацьовано поняття перевантажених операторів. Покращено навички написання коду на мові програмування С++.