**Лабораторна робота №2**

**Тема.** Перевантаження операцій.

**Мета.** Одержати практичні навички створення абстрактних типів даних і перевантаження операцій у мові С++.

**Постановка задачі:** Розширити функціональність класу, реалізовану в попередній роботі, згідно тогож варіанту.

Завдання:

В клас Rectangle додати:

* + Перевантаження:
    - операції ++ (--): одночасно збільшує (зменшує) значення полів а і b;
    - констант true і false: звернення до екземпляра класу дає значення true, якщо прямокутник із заданими довжинами сторін є квадратом, інакше false;
    - операції \*: одночасно домножает поля а і b на скаляр;
    - перетворення типу Rectangle в string (і навпаки).

Головна програма:

Rectangle.h:

#pragma once

#include <iostream>

#include <string>

#include <iomanip>

using namespace std;

class Rectangle

{

private:

double a, b;

public:

Rectangle();

Rectangle(double a, double b);

double P() const;

double S() const;

void setSizes();

void setA();

void setB();

void getSizes() const;

double getA() const;

double getB() const;

bool IsSquare() const;

void WhileNotNormal(double& num);

Rectangle& operator++();

Rectangle& operator--();

Rectangle operator++(int);

Rectangle operator--(int);

operator bool() const;

Rectangle operator\*(int scalar) const;

operator string() const;

Rectangle operator+(const Rectangle& r2) const;

Rectangle operator+(int side) const;

friend Rectangle operator+(int side, const Rectangle& r2);

Rectangle(const string& str);

friend ostream& operator<<(ostream& out, const Rectangle& r);

static void Show();

friend istream& operator>>(istream& in, Rectangle& r);

double& operator[](int index);

Rectangle& operator +=(double num);

Rectangle& operator +=(int num);

Rectangle& operator +=(const Rectangle& r);

friend bool operator ==(const Rectangle& r1, const Rectangle& r2);

friend bool operator >=(const Rectangle& r1, const Rectangle& r2);

friend bool operator <=(const Rectangle& r1, const Rectangle& r2);

friend bool operator >(const Rectangle& r1, const Rectangle& r2);

friend bool operator <(const Rectangle& r1, const Rectangle& r2);

friend bool operator !=(const Rectangle& r1, const Rectangle& r2);

Rectangle& operator =(const Rectangle& r);

};

Rectangle.cpp:

#include "Rectangle.h"

void Rectangle::WhileNotNormal(double& num) {

while (num <= 0) {

cout << "Невiрно задано сторону введiть знову: ";

cin >> num;

}

}

Rectangle::Rectangle(double a, double b) {

WhileNotNormal(a);

WhileNotNormal(b);

this->a = a;

this->b = b;

}

Rectangle::Rectangle() { a = 0; b = 0; }

Rectangle::Rectangle(const string& str) {

int cPos = str.find("|");

if (cPos != string::npos) {

string num1 = str.substr(0, cPos);

string num2 = str.substr(cPos + 1);

a = stod(num1);

b = stod(num2);

}

else

a = b = 0;

}

void Rectangle::Show()

{

cout << setw(15) << "1 сторона" << setw(5) << "|" << setw(15) << "2 сторона" << endl;

}

double Rectangle::P() const {

if (a == 0 || b == 0) {

cout << "Недостатньо значень сторiн для периметру" << endl;

return 0;

}

return 2 \* (a + b);

}

double Rectangle::S() const {

if (a == 0 || b == 0) {

cout << "Недостатньо значень сторiн для площi." << endl;

return 0;

}

return a \* b;

}

void Rectangle::setA() {

cout << "Введiть сторону a: ";

cin >> this->a;

WhileNotNormal(a);

}

void Rectangle::setB() {

cout << "Введiть сторону b: ";

cin >> this->b;

WhileNotNormal(b);

}

void Rectangle::setSizes() {

this->setA();

this->setB();

}

double Rectangle::getA() const {

return a;

}

double Rectangle::getB() const {

return b;

}

void Rectangle::getSizes() const {

if (IsSquare())

cout << "Квадрат\n";

else

cout << "Прямокуник\n";

cout << "Довжина сторони a: " << getA() << ", дожина сторни b:" << getB() << endl;

}

bool Rectangle::IsSquare() const {

if (a == 0 || b == 0) {

cout << "Недостатньо значень сторiн для визначення того чи квадрат." << endl;

return false;

}

return a == b;

}

Rectangle& Rectangle::operator++() {

a += 1.0; b += 1.0;

return \*this;

}

Rectangle& Rectangle::operator--() {

if (a > 1.0 && b > 1.0) {

a -= 1; b -= 1;

}

else

throw exception("Вихiд за межi сторiн");

return \*this;

}

Rectangle Rectangle::operator++(int) {

Rectangle temp = \*this;

a += 1; b += 1;

return temp;

}

Rectangle Rectangle::operator--(int) {

Rectangle temp = \*this;

if (a > 1.0 && b > 1.0) {

a -= 1; b -= 1;

}

else

throw exception("Вихiд за можлве значення");

return temp;

}

Rectangle::operator bool() const { return a == b && a != 0; }

Rectangle::operator string() const {

string num = to\_string(a) + "|" + to\_string(b);

return num;

}

Rectangle Rectangle::operator+(const Rectangle& r2) const

{

return Rectangle(a + r2.a, b + r2.b);

}

Rectangle Rectangle::operator+(int side) const

{

return Rectangle(a + side, b + side);

}

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

double x1 = a, x2 = b;

if (scalar > 0) {

x1 \*= scalar;

x2 \*= scalar;

}

else throw exception("Не можна множити на вiд'ємну сторону");

return Rectangle(x1, x2);

}

Rectangle operator+(int side, const Rectangle& r2)

{

return Rectangle(r2.a + side, r2.b + side);

}

double& Rectangle::operator [](int index)

{

if (index == 0)

return a;

else if (index == 1)

return b;

else

throw out\_of\_range ("Невiрний iндекс");

}

Rectangle& Rectangle::operator+=(double num)

{

a += num; b += num;

return \*this;

}

Rectangle& Rectangle::operator+=(int num)

{

a += num; b += num;

return \*this;

}

Rectangle& Rectangle::operator+=(const Rectangle& r)

{

a += r.a;

b += r.b;

return \*this;

}

ostream& operator<<(ostream& out, const Rectangle& r) {

out << setw(15) << r.a << setw(5) << "|" << setw(15) << r.b;

return out;

}

istream& operator>>(istream& in, Rectangle& r)

{

double x1 = -1, x2 = -1;

while (x1 <= 0 || x2 <= 0 )

if (!(in >> x1 && in >> x2))

return in;

r.a = x1;

r.b = x2;

return in;

}

bool operator==(const Rectangle& r1, const Rectangle& r2)

{

return r1.a == r2.a && r1.b == r2.b;

}

bool operator>=(const Rectangle& r1, const Rectangle& r2)

{

return r1.S() >= r2.S();

}

bool operator<=(const Rectangle& r1, const Rectangle& r2)

{

return r1.S() <= r2.S();

}

bool operator>(const Rectangle& r1, const Rectangle& r2)

{

return !(r1 <= r2);

}

bool operator<(const Rectangle& r1, const Rectangle& r2)

{

return !(r1 >= r2);

}

bool operator!=(const Rectangle& r1, const Rectangle& r2)

{

return !(r1 == r2);

}

Rectangle& Rectangle::operator=(const Rectangle& r) {

if (\*this == r)

return \*this;

a = r.a;

b = r.b;

return \*this;

}

Main.cpp:

#include "Rectangle.h"

int main() {

setlocale(0, "UKR");

//Основні елементи

Rectangle rect = Rectangle(5, 5);

string r1STR = rect;

cout << "Str = " << r1STR << endl;

Rectangle::Show();

cout << rect << "(Об'єкт rect - початкове значення)" << endl;

cout << rect++ << "(rect++)" << endl;

cout << rect-- << "(rect--)"<< endl;

cout << ++rect << "(++rect)" << endl;

cout << --rect << "(--rect)" << endl;

cout << "rect\*5: " << endl;

Rectangle::Show();

cout << rect \* 5 << endl;

cout << endl;

cout << "Рядкове значення (r):" << endl;

Rectangle r = r1STR;

cout << "r квадрат? ";

if (r)

cout << "Квадрат" << endl;

Rectangle::Show();

cout << r << endl;

Rectangle rectangle1 = r1STR;

Rectangle rectangle2(10, 10);

cout << endl;

Rectangle::Show();

cout << rectangle1 << " (rectngle1 початкове значення)" << endl;

cout << rectangle2 << " (rectngle2 почакове значення)" << endl;

cout << rectangle1 + rectangle2 << " (Сума двох прямокутникiв)"<<endl;

cout << 10 + rectangle1 << " (Число + прямокутник:)" << endl;

cout << rectangle1 + 10 << " (Прямокутник + число)" << endl;

//Індексація та +=

Rectangle rr;

Rectangle::Show();

cout << rr << "(пустий об'єкт rr)" << endl;

cout << "Введiть значення для полiв Rectangle (Перша сторона, друга сторона): "; cin >> rr;

cout << rr << "(Об'єкт rr піля введення)" << endl;

(rr += 5)+=10;

cout << rr << "Об'єкт rr пiсля операцiї (rr += 5)+=10" << endl;

cout << "Доступ до полiв за iндексом: " << endl;

cout << "rr[0] = " << rr[0] << endl;

rr[0] = 10;

cout << "rr[0] має дорiвнювати 10: " << rr[0] << endl;

//Логіні операції

Rectangle r1(10, 20), r2(5, 10);

Rectangle::Show();

cout << r1 << "(Об'єкт r1)" << endl;

cout << r2 << "(Об'єкт r2)" << endl;

bool a = r1 == r2;

cout << "r1 == r2 " << a << endl;

a = r1 >= r2;

cout << "r1 >= r2 " << a << endl;

a = r1 > r2;

cout << "r1 > r2 " << a << endl;

a = r1 <= r2;

cout << "r1 <= r2 " << a << endl;

a = r1 < r2;

cout << "r1 < r2 " << a << endl;

a = r1 != r2;

cout << "r1 != r2 " << a << endl;

// Перевизначені оператори потоку

cout << "Створено масив об'єктiв:" << endl;

Rectangle rectangles[5];

for (int i = 0; i < 5; i++) {

cout << "Введiть " << i + 1 << " прямокутник: ";

cin >> rectangles[i];

}

Rectangle::Show();

for (int i = 0; i < 5; i++)

cout << rectangles[i] << endl;

}

Тестовий набір

|  |  |  |
| --- | --- | --- |
| № | Програма | Тествоий набір |
|  |  |  |