# КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ТАРАСА ШЕВЧЕНКА



## ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

**Кафедра прикладних інформаційних систем**

**Звіт до лабораторної роботи №1**

# з курсу

**«Об’єктно-орієнтоване програмування»**

*Студента 2 курсу*

*групи ПП-11 спеціальності 122 «Комп'ютерні науки» ОП «Прикладне програмування»*

%username%

*Викладач:*

к.ф.-м.н., доц. Шолохов О.В.

## Київ – 202

**1.Назва роботи**

Успадкування в С++ та С#

1. **Тема роботи**

Успадкування в С++ та С#

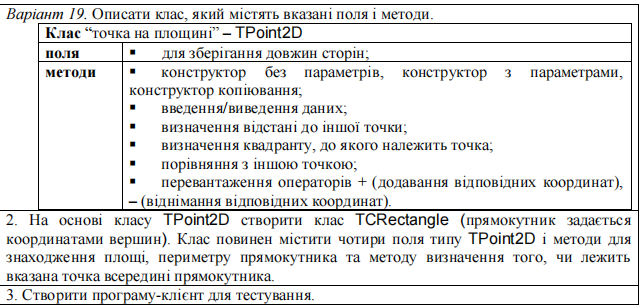
1. **Мета роботи**

Засвоєння теоретичного матеріалу та набуття практичного досвіду

використання успадкування в С++ та С# при складанні різних алгоритмів.

1. **Умова завдання**

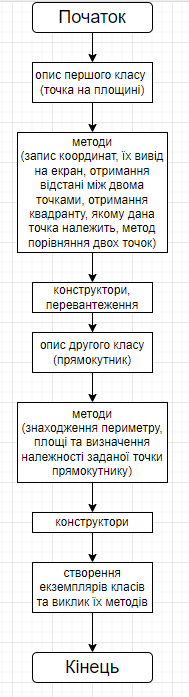
(19 варіант)



1. **Рішення**

Описуємо перший клас, що являє собою точку на площині (має два числових поля для х та у координат). Описуємо його методи (запис координат, їх вивід на екран, отримання відстані між двома точками за формулою кореню з суми квадратів різниць відповідних координат, отримання квадранту методом порівняння координат точки з границями координатної площини, метод порівняння двох точок). Оголошуємо перевантаження операторів “+” та “-” для додавання та віднімання координат відповідно. Описуємо конструктори класу: конструктор без параметрів, конструктор з параметрами, конструктор копіювання. Після цього описуємо другий клас, що є прямокутником та наслідує клас точки: в ньому оголошуємо чотири поля вищевказаних точок для зберігання вершин, а також методи для знаходження периметру, площі та визначення належності заданої точки прямокутнику. Створюємо екземпляри класів та викликаємо потрібні методи.

Блок-схема:



Код С#:

using System;

namespace Ac\_227

{

class Program

{

public class Point2D

{

public int x;

public int y;

public void setCoordinates()

{

Console.WriteLine("Please enter X:");

x = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Please enter Y:");

y = Convert.ToInt32(Console.ReadLine());

}

public void writeCoordinates()

{

Console.WriteLine("X is: " + x + "; Y is: " + y);

}

public static int getDistance(Point2D firstPoint, Point2D otherPoint)

{

int dist = (int)Math.Sqrt(Math.Pow((otherPoint.x - firstPoint.x), 2) + Math.Pow((otherPoint.y - firstPoint.y), 2));

Console.WriteLine("Distance between two points is: " + dist);

return dist;

}

public void getQuadrant()

{

if (x > 0 && x < 100 && y > 0 && y < 100)

{

Console.WriteLine("This point belongs to second quadrant");

} else if (x > 0 && x < 100 && y < 0 && y > -100)

{

Console.WriteLine("This point belongs to third quadrant");

} else if (x > -100 && x < 0 && y < 0 && y > -100)

{

Console.WriteLine("This point belongs to fourth quadrant");

} else

{

Console.WriteLine("This point belongs to first quadrant");

}

}

public void comparePoints(Point2D firstPoint, Point2D otherPoint)

{

if (otherPoint.x == firstPoint.x && otherPoint.y == firstPoint.y)

{

Console.WriteLine("These points are equal");

} else

{

Console.WriteLine("These points are not equal; they have " + getDistance(firstPoint, otherPoint) + " units between them.");

}

}

public static Point2D operator+ (Point2D firstPoint, Point2D secondPoint)

{

Point2D result = new Point2D();

result.x = firstPoint.x + secondPoint.x;

result.y = firstPoint.y + secondPoint.y;

return result;

}

public static Point2D operator- (Point2D firstPoint, Point2D secondPoint)

{

Point2D result = new Point2D();

result.x = firstPoint.x - secondPoint.x;

result.y = firstPoint.y - secondPoint.y;

return result;

}

public Point2D()

{

x = 0;

y = 0;

}

public Point2D(int xInput, int yInput)

{

x = xInput;

y = yInput;

}

public Point2D(Point2D givenPoint)

{

x = givenPoint.x;

y = givenPoint.y;

}

}

public class Rectangle : Point2D

{

Point2D one;

Point2D two;

Point2D three;

Point2D four;

public void getPerimeter()

{

int oneTwo = Point2D.getDistance(one, two);

int twoThree = Point2D.getDistance(two, three);

int threeFour = Point2D.getDistance(three, four);

int fourOne = Point2D.getDistance(four, one);

Console.WriteLine("Perimeter is " + (oneTwo + twoThree + threeFour + fourOne) + " units long.");

}

public void getArea()

{

int oneTwo = Point2D.getDistance(one, two);

int twoThree = Point2D.getDistance(two, three);

Console.WriteLine("Area is " + oneTwo \* twoThree + " units.");

}

public void doesPointBelong(Point2D possiblyACreep)

{

if ((possiblyACreep.x >= one.x) && (possiblyACreep.x <= three.x) && (possiblyACreep.y >= three.y) && (possiblyACreep.y <= one.y))

{

Console.WriteLine("Point belongs.");

} else

{

Console.WriteLine("Point does not belong here.");

}

}

public Rectangle(Point2D oneInput, Point2D twoInput, Point2D threeInput, Point2D fourInput)

{

one = oneInput;

two = twoInput;

three = threeInput;

four = fourInput;

}

}

static void Main(string[] args)

{

Point2D one = new Point2D(-50, 50);

Point2D two = new Point2D(50, 50);

Point2D three = new Point2D(50, -50);

Point2D four = new Point2D(-50, -50);

Point2D test = new Point2D();

one.getQuadrant();

two.getQuadrant();

three.getQuadrant();

four.getQuadrant();

one.comparePoints(one, two);

test.setCoordinates();

Point2D overload = one + two;

overload.writeCoordinates();

Rectangle thisIsARectangle = new Rectangle(one, two, three, four);

thisIsARectangle.getArea();

thisIsARectangle.getPerimeter();

thisIsARectangle.doesPointBelong(test);

}

}

}

Код на С++:

#include <iostream>

#include <iomanip>

#include <math.h>

using namespace std;

int main()

{

class Point2D

{

public: int x; int y;

void setCoordinates()

{

cout << "Please enter X:" << endl;

cin >> x;

cout << "Please enter Y:" << endl;

cin >> y;

}

void writeCoordinates()

{

cout << "X is: " << x << "; Y is: " << y << endl;

}

int getDistance(Point2D firstPoint, Point2D otherPoint)

{

int dist = sqrt(pow((otherPoint.x - firstPoint.x), 2) + pow((otherPoint.y - firstPoint.y), 2));

return dist;

}

void printDistance(Point2D firstPoint, Point2D otherPoint) {

int dist = sqrt(pow((otherPoint.x - firstPoint.x), 2) + pow((otherPoint.y - firstPoint.y), 2));

cout << "Distance between two points is: " << dist << endl;

}

void getQuadrant()

{

if (x > 0 && x < 100 && y > 0 && y < 100)

{

cout << "This point belongs to second quadrant" << endl;

}

else if (x > 0 && x < 100 && y < 0 && y > -100)

{

cout << "This point belongs to third quadrant" << endl;

}

else if (x > -100 && x < 0 && y < 0 && y > -100)

{

cout << "This point belongs to fourth quadrant" << endl;

}

else

{

cout << "This point belongs to first quadrant" << endl;

}

}

void comparePoints(Point2D firstPoint, Point2D otherPoint)

{

if (otherPoint.x == firstPoint.x && otherPoint.y == firstPoint.y)

{

cout << "These points are equal" << endl;

}

else

{

cout << "These points are not equal; they have " << getDistance(firstPoint, otherPoint) << " units between them." << endl;

}

}

Point2D operator+(const Point2D secondPoint) {

Point2D result = Point2D();

result.x = this->x + secondPoint.x;

result.y = this->y + secondPoint.y;

return result;

}

Point2D operator- (const Point2D secondPoint)

{

Point2D result = Point2D();

result.x = this->x - secondPoint.x;

result.y = this->y - secondPoint.y;

return result;

}

Point2D()

{

x = 0;

y = 0;

}

Point2D(int xInput, int yInput)

{

x = xInput;

y = yInput;

}

Point2D(const Point2D& givenPoint)

{

x = givenPoint.x;

y = givenPoint.y;

}

};

class Rectangle : public Point2D

{

public:

Point2D one;

Point2D two;

Point2D three;

Point2D four;

void getPerimeter()

{

int oneTwo = Point2D::getDistance(one, two);

int twoThree = Point2D::getDistance(two, three);

int threeFour = Point2D::getDistance(three, four);

int fourOne = Point2D::getDistance(four, one);

cout << "Perimeter is " << (oneTwo + twoThree + threeFour + fourOne) << " units long." << endl;

}

void getArea()

{

int oneTwo = Point2D::getDistance(one, two);

int twoThree = Point2D::getDistance(two, three);

cout << "Area is " << (oneTwo \* twoThree) << " units." << endl;

}

void doesPointBelong(Point2D possiblyACreep)

{

if ((possiblyACreep.x >= one.x) && (possiblyACreep.x <= three.x) && (possiblyACreep.y >= three.y) && (possiblyACreep.y <= one.y))

{

cout << "Point belongs." << endl;

}

else

{

cout << "Point does not belong here." << endl;

}

}

Rectangle(Point2D oneInput, Point2D twoInput, Point2D threeInput, Point2D fourInput)

{

one = oneInput;

two = twoInput;

three = threeInput;

four = fourInput;

}

};

Point2D one = Point2D(-50, 50);

Point2D two = Point2D(50, 50);

Point2D three = Point2D(50, -50);

Point2D four = Point2D(-50, -50);

Point2D test = Point2D();

one.getQuadrant();

two.getQuadrant();

three.getQuadrant();

four.getQuadrant();

one.comparePoints(one, two);

test.setCoordinates();

one + two;

cout << "Overload results:" << endl;

one.writeCoordinates();

Rectangle thisIsARectangle = Rectangle(one, two, three, four);

thisIsARectangle.getArea();

thisIsARectangle.getPerimeter();

thisIsARectangle.doesPointBelong(test);

}

Результат роботи програми C#:

[screenshot]

Результат роботи програми С++:

[screenshot]

**Контрольні питання**

1. У чому полягає суть успадкування?

Класи-нащадки отримують методи та поля класів-предків без перезапису

1. Що таке клас-предок?

Клас, який наслідують (передає методи/поля)

1. Що таке клас-нащадок?

Клас, який наслідує (отримує методи/поля)

1. Для чого використовують класи-предки?

Щоб не створювати класи, що мають спільні поля/методи і тим самим економити ресурси та час

1. Скільки класів-предків може мати клас?

B C# та Java лише один, тоді як у С++ існує можливість множинного наслідування (однак, вона може призвести до так званої “проблеми діаманту”: два нащадки певного класу мають ще одного спільного предка)

IMG_256

1. Як описати клас-нащадок?

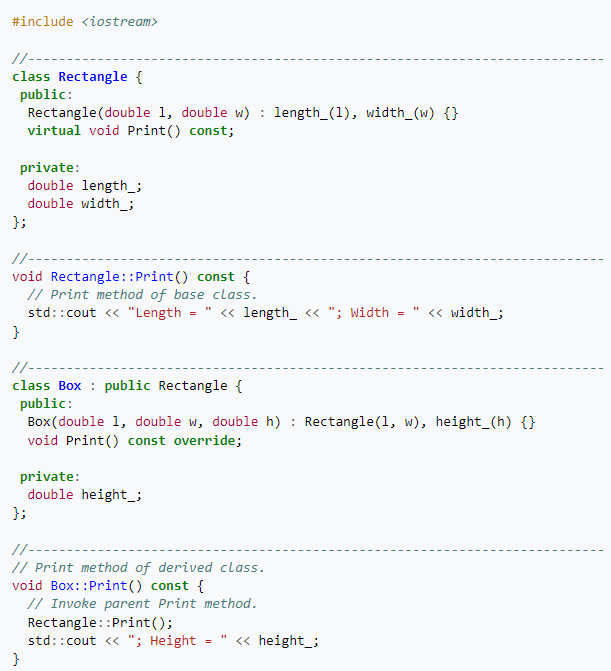
class Child : Parent {…}

1. Що таке перевизначення методів?

Можливість створення метода, аналогічного методу предка, з однаковими параметрами та типом значення, що повертається, але з іншою реалізацією.

1. Як перевизначити метод класу-предка?

Перевизначення в С++ потребує наступного синтаксису: перед ім’ям метода в обох класах записують їх назви з двома двокрапками.



Перевизначення в С# має таку ж логіку, але перед методом класа-предка пишемо модифікатор virtual, а перед методом класа-нащадка - override.

1. Як здійснити доступ до перевизначених методів класу-предка?

Викликати даний метод в контексті об’єкту класу-предка.

1. Чи має об’єкт-нащадок доступ до перевизначених методів класу-предка?

Ні, не має

1. Які члени класу-предка є доступними для класу-нащадка?

public та protected

1. Чи має можливість клас-нащадок змінювати область видимості членів класу-предка?

Ні, може змінювати лише область видимості своїх членів.

13. Як описати члени класу-предка, щоб вони були доступними для класу-нащадка і

недоступними у програмі-клієнті?

Використовувати модифікатор protected.

1. Як описати члени класу-предка, щоб вони не були доступними як для класу-нащадка, так і для програми-клієнта?

Використовувати модифікатор private.

1. Чи може змінна типу класу-предка приймати значення об’єкта-нащадка?

Так, може.

1. До яких полів і методів об’єкта-нащадка може здійснити доступ об’єкт-предок?

Лише до перевизначених.

1. **Висновки**

В результаті виконання даної лабораторної роботи я оволодів практичними навичками використання успадкування в С#, навчився складати програми для виконання операцій з використанням успадкування, перевантаження операторів. Вважаю дану лабораторну роботу виконаною в повному обсязі.