**ПРАКТИЧНЕ ЗАВДАННЯ № 8**

**Завдання №1 :**

Добавьте недостающие оператор-функции:

Перегруженный оператор + должен поэлементно складывать оба операнда. Перегруженный оператор — должен вычитать все элементы правого операнда из элементов левого. Перегруженный оператор == должен возвращать значение true, если все элементы обоих операндов равны, в противном случае он должен возвращать значение false.

**Код програми:**

#include <iostream >

using namespace std;

class Array {

int nums[10];

public:

Array();

void set(int n[10]);

void show();

Array operator +(Array obj2);

Array operator -(Array obj2);

int operator ==(Array obj2);

};

Array::Array() {

for (int i = 0; i < 10; ++i)

nums[i] = 0;

}

void Array::set(int \*n) {

for (int i = 0; i < 10; ++i)

nums[i] = n[i];

}

void Array::show() {

for (int i = 0; i < 10; ++i)

cout << nums[i] << ' ';

cout << "\n";

}

Array Array::operator +(Array obj2){

Array temp;

for(int i = 0; i < 10; ++i)

temp.nums[i] = nums[i] + obj2.nums[i];

return temp;

}

Array Array::operator -(Array obj2){

Array temp;

for(int i = 0; i < 10; ++i)

temp.nums[i] = nums[i] - obj2.nums[i];

return temp;

}

int Array::operator ==(Array obj2){

for(int i = 0; i < 10; ++i){

if(obj2.nums[i] != nums[i])

return 0;

}

return 1;

}

int main() {

Array obj1, obj2, obj3;

int i[10] = { 1, 2, 3, 4, 5, 6, 7, 8 ,9 ,10 };

obj1.set(i);

obj2.set(i);

obj1.show();

obj2.show();

obj3 = obj1 + obj2;

obj3.show();

obj3 = obj1 - obj3;

obj3.show();

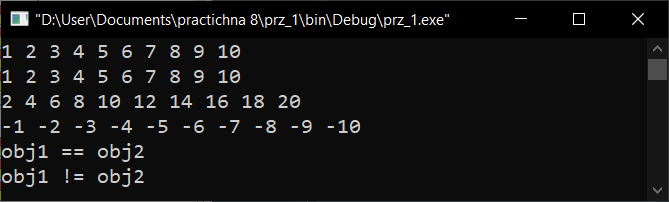
cout << ((obj1 == obj2) ? "obj1 == obj2\n" : "obj1 != obj2\n");

cout << ((obj1 == obj3) ? "obj1 == obj2\n" : "obj1 != obj2\n");

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №2 :**

Переработайте программу Задача 8.1 так, чтобы перегрузить операторы с использованием дружественных функций.

**Код програми:**

#include <iostream >

using namespace std;

class Array {

int array[10];

public:

Array();

void set(int n[10]);

void show();

friend Array operator +(Array &obj1, Array &obj2);

friend Array operator -(Array &obj1, Array &obj2);

friend int operator ==(Array &obj1, Array &obj2);

};

Array::Array() {

for (int i = 0; i < 10; ++i) array[i] = 0;

}

void Array::set(int \*n) {

for (int i = 0; i < 10; ++i) array[i] = n[i];

}

void Array::show() {

for (int i = 0; i < 10; ++i)

cout << array[i] << ' ';

cout << "\n";

}

Array operator +(Array &obj1, Array &obj2){

Array temp;

for(int i = 0; i < 10; ++i)

temp.array[i] = obj1.array[i] + obj2.array[i];

return temp;

}

Array operator -(Array &obj1, Array &obj2){

Array temp;

for(int i = 0; i < 10; ++i)

temp.array[i] = obj1.array[i] - obj2.array[i];

return temp;

}

int operator ==(Array &obj1, Array &obj2){

for(int i = 0; i < 10; ++i)

return (obj1.array[i] == obj2.array[i]) ? 1 : 0;

}

int main() {

Array obj1, obj2, obj3;

int i[10] = { 1, 2, 3, 4, 5, 6, 7, 8 ,9 ,10 };

obj1.set(i);

obj2.set(i);

obj1.show();

obj2.show();

obj3 = obj1 + obj2;

obj3.show();

obj3 = obj1 - obj3;

obj3.show();

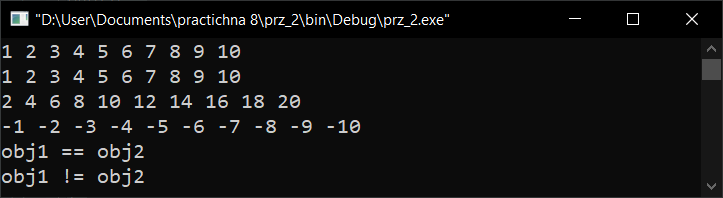
cout << ((obj1 == obj2) ? "obj1 == obj2\n" : "obj1 != obj2\n");

cout << ((obj1 == obj3) ? "obj1 == obj2\n" : "obj1 != obj2\n");

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №3 :**

Используя класс и функции Задача 8.1, перегрузите оператор ++ с помощью функции-члена класса, а оператор — с помощью дружест венной функции. (Перегрузите только префиксные формы операторов ++ и —).

**Код програми:**

#include <iostream >

using namespace std;

class Array {

int array[10];

public:

Array();

void set(int n[10]);

void show();

friend Array operator +(Array &obj1, Array &obj2);

friend Array operator -(Array &obj1, Array &obj2);

friend int operator ==(Array &obj1, Array &obj2);

Array operator ++();

friend Array operator --(Array &obj1);

};

Array::Array() {

for (int i = 0; i < 10; ++i) array[i] = 0;

}

void Array::set(int \*n) {

for (int i = 0; i < 10; ++i) array[i] = n[i];

}

void Array::show() {

for (int i = 0; i < 10; ++i)

cout << array[i] << ' ';

cout << "\n";

}

Array operator +(Array &obj1, Array &obj2){

Array temp;

for(int i = 0; i < 10; ++i)

temp.array[i] = obj1.array[i] + obj2.array[i];

return temp;

}

Array operator -(Array &obj1, Array &obj2){

Array temp;

for(int i = 0; i < 10; ++i)

temp.array[i] = obj1.array[i] - obj2.array[i];

return temp;

}

int operator ==(Array &obj1, Array &obj2){

for(int i = 0; i < 10; ++i)

return (obj1.array[i] == obj2.array[i]) ? 1 : 0;

}

Array Array::operator ++(){

for(int i = 0; i < 10; ++i)

array[i]++;

return \*this;

}

Array operator-- (Array &obj1){

for(int i = 0; i < 10; ++i)

obj1.array[i]--;

return obj1;

}

int main() {

Array obj1, obj2, obj3;

int i[10] = { 1, 2, 3, 4, 5, 6, 7, 8 ,9 ,10 };

obj1.set(i);

obj2.set(i);

obj1.show();

obj2.show();

obj3 = obj1 + obj2;

obj3.show();

obj3 = obj1 - obj3;

obj3.show();

++obj3;

obj3.show();

--obj3;

obj3.show();

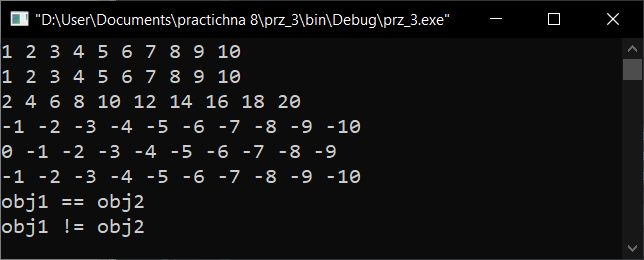
cout << ((obj1 == obj2) ? "obj1 == obj2\n" : "obj1 != obj2\n");

cout << ((obj1 == obj3) ? "obj1 == obj2\n" : "obj1 != obj2\n");

return 0;

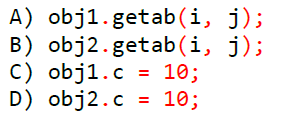
}

**Результати роботи програми (скриншоти)**



**Завдання №4 :**

Какая из следующих инструкций правильна внутри функции main() и почему?



**Код програми:**

#include <iostream>

using namespace std;

class mybase {

int a, b;

public:

int c;

void setab(int i, int j) { a = i; b = j; }

void getab(int &i, int &j) { i = a; j = b; }

};

class derived1 : public mybase {

// ...

};

class derived2 : private mybase {

// ...

};

int main() {

derived1 obj1;

derived2 obj2;

int i, j;

// ...

}

/\*A & C.

Тому що клас derived1 наслідує базовий як public, тобто для нього відкриті всі члени базового класу визначені як public і матиме доступ до них поза класом. Клас derived2 наслідується як private, тому для нього всі відкриті члени стають закритими і можуть бути доступні лише через функції-члени дочірного класу\*/

**Завдання №5 :**

Если бы переменные а и b внутри класса myclass Задача 8.4 стали не закрытыми (по умолчанию), а защищенными членами, изменился бы какой-нибудь из ваших ответов на вопросы этого упражнения? Если да, то почему?

**Код програми:**

/\*Так само A & C. Тому що специфікатор ptotected дозволяє деяким членам (базового) класу залишитись захищеними. Отже, вони мають такі ж особливості доступу, як і protected, тому і доступні лише функціям-членам дочірних класів.\*/

**Завдання №6 :**

В приведенном ниже фрагменте добавьте конструктор для класса myderived. Он должен передать указатель на инициализируемую строку конструктору класса mybase. Кроме того, конструктор myderived() должен инициализировать переменную len длиной строки.

**Код програми:**

#include <iostream>

#include <cstring>

using namespace std;

class mybase {

char str[80];

public:

mybase(char \*s) { strcpy (str, s); }

char \*get() { return str; }

};

class myderived : public mybase {

int len;

public:

myderived (char \*n) : mybase(n){ len = strlen(n); }

int getlen() { return len; }

void show() { cout << get () << endl; }

};

int main() {

myderived obj((char \*)"hello");

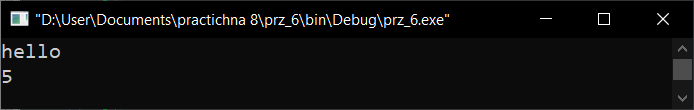
obj.show();

cout << obj.getlen() << endl;

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №7 :**

Используя следующий фрагмент, создайте соответствующие конструкторы саг() и truck(). Они должны передавать необходимые аргументы объектам класса vehicle. Кроме этого конструктор саг() должен при создании объекта инициализировать переменную passengers, а конструктор truck() - переменную loadlimit.

Для конструкторов саг() и truck() объекты должны объявляться следующим образом:

car obj(passengers, wheels, range);

truck obj(loadlimit, wheels, range);

**Код програми:**

#include <iostream>

using namespace std;

class vehicle {

int num\_wheels;

int range;

public:

vehicle(int w, int r) {

num\_wheels = w;

range = r;

}

void showv() {

cout << "Wheels: " << num\_wheels << '\n';

cout << "Range: " << range << '\n';

}

};

class car : public vehicle {

int passengers;

public:

car(int p, int w, int r): vehicle(w, r) {

passengers = p;

}

void show() {

showv();

cout << "Passengers: " << passengers << '\n';

}

};

class truck : public vehicle {

int loadlimit;

public:

truck(int l, int w, int r):vehicle(w, r){

loadlimit = l;

}

void show() {

showv();

cout << "Loadlimit: " << loadlimit << '\n';

}

};

int main() {

car objc(5, 4, 500);

truck objt(3000, 12, 1200);

cout << "Car:\n";

objc.show();

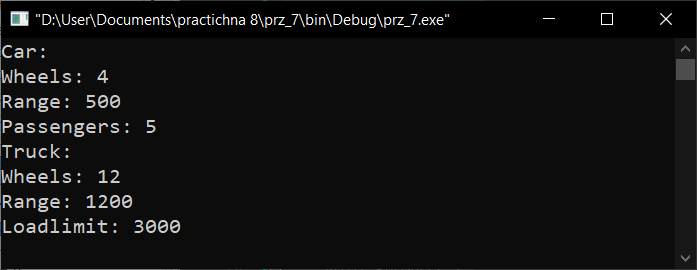
cout << "Truck:\n";

objt.show();

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №8 :**

Используя следующую иерархию классов, создайте конструктор класса С так, чтобы он инициализировал переменную k и передавал аргументы конструкторам А() и В().

**Код програми:**

#include <iostream>

using namespace std;

class A {

int i;

public:

A(int a) { i = a; }

void show\_a(){ cout << " Class A a = " << i << endl; }

};

class B {

int j;

public:

B(int a) { j = a; }

void show\_b(){

cout << " Class B b = " << j << endl;

}

};

class C : public A, public B {

int k;

public:

C(int a, int b, int c): A(a), B(b) { k = c; }

void show(){

cout << endl;

show\_a();

show\_b();

cout << " Class C c = " << k << endl;

}

};

int main(){

A a(4);

a.show\_a();

B b(5);

b.show\_b();

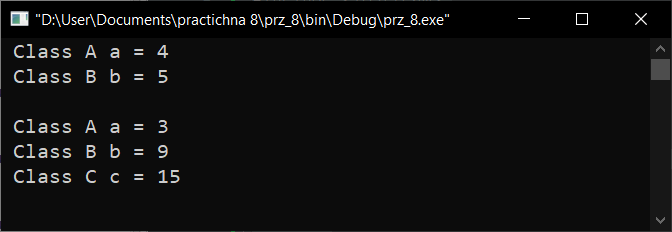
C c(3, 9, 15);

c.show();

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №9 :**

Создайте исходный базовый класс building для хранения числа этажей и комнат в здании, а также общую площадь комнат. Создайте производный класс house, который наследует класс building и хранит число ванных комнат и число спален. Кроме этого создайте производный класс office, который наследует класс building и хранит число огнетушителей и телефонов.

**A)**

**Код програми:**

#include <iostream>

using namespace std;

class building {

int floors;

int rooms;

double square;

public:

building(int f, int r, double s){

floors = f;

rooms = r;

square = s;

}

void show\_building(){

cout << "floors = " << floors << " rooms = " << rooms << " square = " << square << endl;

}

};

class house : public building {

int bathrooms;

int bedrooms;

public:

house(int f, int r, double s, int bath, int bed): building(f, r, s) {

bathrooms = bath;

bedrooms = bed;

}

void show\_house(){

cout << endl << "house: " << endl;;

show\_building();

cout << "bathrooms = " << bathrooms << " bedrooms = " << bedrooms << endl;

}

};

class office : public building {

int extinguisher;

int phones;

public:

office(int f, int r, double s, int ex, int ph): building(f, r, s) {

extinguisher = ex;

phones = ph;

}

void show\_office(){

cout << endl << "office " << endl;

show\_building();

cout << "extinguisher = " << extinguisher << " phones = " << phones << endl;

}

};

int main() {

house h(10, 35, 345.452, 15, 19);

office f(25, 136, 1244.1234, 45, 224);

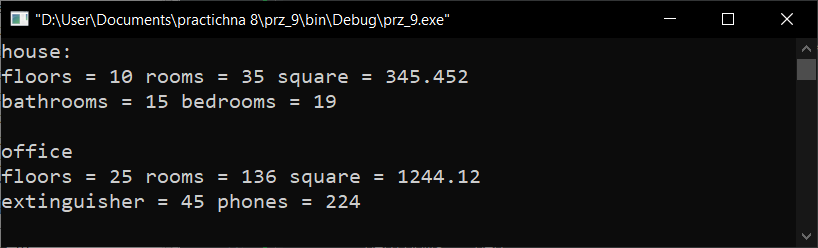
h.show\_house();

f.show\_office();

return 0;

}

**Результати роботи програми (скриншоти)**



**B)**

**Код програми:**

#include <iostream>

using namespace std;

class Building {

protected:

int floors;

int rooms;

double square;

};

class House : public Building {

int bathrooms;

int bedrooms;

public:

House(int f, int r, double s, int bath, int bed) {

floors = f;

rooms = r;

square = s;

bathrooms = bath;

bedrooms = bed;

}

void show() {

cout << "House: " << endl;;

cout << "floors = " << floors << " rooms = " << rooms << " square = " << square << endl;

cout << "bathrooms = " << bathrooms << " bedrooms = " << bedrooms << endl;

}

};

class Office : public Building {

int extinguisher;

int phones;

public:

Office(int f, int r, double s, int ex, int ph) {

floors = f;

rooms = r;

square = s;

extinguisher = ex;

phones = ph;

}

void show(){

cout << endl << "Office " << endl;

cout << "floors = " << floors << " rooms = " << rooms << " square = " << square << endl;

cout << "extinguisher = " << extinguisher << " phones = " << phones << endl;

}

};

int main() {

House h(10, 35, 345.452, 15, 19);

Office f(25, 136, 1244.1234, 45, 224);

h.show();

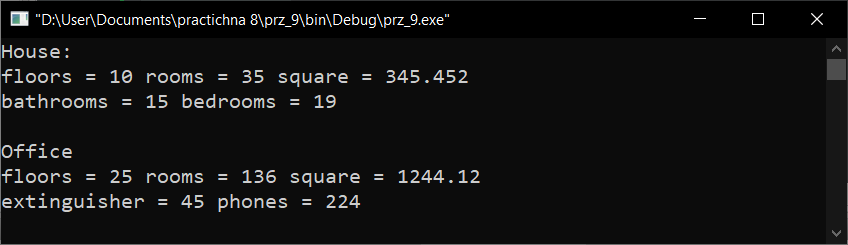
f.show();

return 0;

}

/\*В другій версії програми творено клас *Building,* який має тільки зберігати змінні *floors, rooms, square,* тому їх позначено специфікатором protected. Адже члени класу позначені як protected вважаються закритими, але також наслідуються дочірним класом: фукції члени дочірного мають доступ до захищених членів базового класу, тобто ми можемо змінювати їх в дочірному класі, але не поза ним\*/

**Результати роботи програми (скриншоти)**



**Завдання №10 :**

Впишите детали, как указано в комментариях:

/\* Create earth(double d, int r). Have it pass the distance and days of revolution back

to planet.

Have it compute the circumference of the orbit. (Hint: circumference = 2r \*3.1416.)\*/

/\* Create a function called show() that displays the information.\*/

**Код програми:**

#include <iostream>

using namespace std;

class planet {

protected:

double distance ; // miles from the sun

int revolve ; // in days

public:

planet(double d, int r) { distance = d; revolve = r; }

};

class earth : public planet {

double circumference; // circumference(окружность) of orbit

public:

earth(double d, int r): planet(d, r) {

circumference = 2 \* r \* 3.1416;

}

void show(){

cout << "distance " << distance << " revolve " << revolve << " circumference " << circumference << endl;

}

};

int main() {

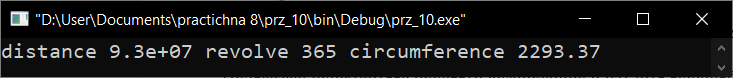
earth obj(93000000, 365);

obj.show();

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №11 :**

Иерархия классов из Задача 8.7 с классом vehicle. В программе имеется ошибка и код не форматирован. Найдите ошибку.

Подсказка: попытайтесь провести компиляцию и изучите сообщения об ошибках.

При компиляции программы вы могли увидеть предупреждающее сообщение (или, возможно, сообщение об ошибке), связанное с использованием инструкции switch внутри классов саг и motorised. Почему?

**Код програми:**

# include <iostream >

using namespace std;

class vehicle {

int num\_wheels ;

int range ;

public :

vehicle (int w, int r){

num\_wheels = w;

range = r;

}

void show\_v () {

cout << " Wheels : " << num\_wheels << '\n'; cout << " Range : " << range << '\n';

}

};

enum motor {gas , electric , diesel };

class motorized : virtual public vehicle {

enum motor mtr ;

public :

motorized ( enum motor m, int w, int r) : vehicle (w, r) {

mtr = m;

}

void showm () {

cout << " Motor : ";

switch (mtr) {

case gas : cout << "Gas \n";

break ;

case electric : cout << " Electric \n";

break ;

case diesel : cout << " Diesel \n";

break ;

}

}

};

class road\_use : virtual public vehicle {

int passengers ;

public :

road\_use (int p, int w, int r) : vehicle (w, r) {

passengers = p;

}

void showr() {

cout << " Passengers : " << passengers << '\n';

}

};

enum steering { power, rack\_pinion, manual };

class car : public motorized, public road\_use {

enum steering strng;

public:

car ( enum steering s, enum motor m, int w, int r, int p) :

road\_use (p, w, r), motorized (m, w, r), vehicle (w, r) {

strng = s;

}

void show () {

show\_v();

showr();

showm();

cout << " Steering : ";

switch (strng) {

case power : cout << " Power \n";

break;

case rack\_pinion : cout << " Rack and Pinion \n";

break;

case manual : cout << " Manual \n";

break;

}

}

};

int main () {

car c(power , gas , 4, 500 , 5);

c. show ();

return 0;

}

/\*Помилка неоднозначністі виникає через при множинному наслідуванні: клас ***car*** наслідує базовий ***vehical*** через 2 дочірних ***motorized*** і ***road\_use***, тобто двічі, через що крмпілятор не знає через який із дочірних наслідувати ***vehical.*** Вирішення проблеми полягає в тому, що потрібно, щоб базовий клас наслідувався як ***virtual*** всіма дочірними:

*class motorized : virtual public vehicle {*

*//…*

*}*

*class road\_use : virtual public vehicle {*

*//…*

*}*

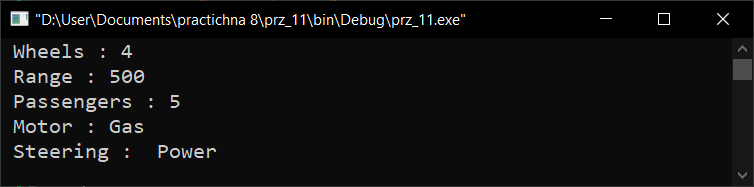
*class car : public motorized, public road\_use {*

*//…*

*}*

*\*/*

**Результати роботи програми (скриншоти):**



**Завдання №12 :**

Здесь представлена переработанная версия класса coord из предыдущего Практического занятия. Форматирование кода нарушено. Теперь этот класс используется в качестве базового для класса quad, в котором помимо координат хранится номер квадранта, к которому принадлежит точка с этими координатами. Запустите программу и попытайтесь понять и объяснить полученный результат.

**Код програми:**

#include <iostream>

using namespace std;

class Coord {

public:

int x, y;

Coord() { x = 0; y = 0; }

Coord(int i, int j) { x = i; y = j; }

void get\_xy(int &i, int &j) { i = x; j = y; }

friend Coord operator +(Coord obj1, Coord obj2);

friend Coord operator -(Coord obj1, Coord obj2);

Coord operator =(Coord ob2);

};

Coord operator +(Coord obj1, Coord obj2) {

Coord temp;

cout << " Using Coord operator +() \n";

temp.x = obj1.x + obj2.x;

temp.y = obj1.y + obj2.y;

return temp;

}

Coord operator -(Coord obj1, Coord obj2) {

Coord temp;

cout << " Using Coord operator -() \n";

temp.x = obj1.x + obj2.x;

temp.y = obj2.y + obj2.y;

return temp;

}

Coord Coord::operator =(Coord ob2) {

cout << " Using Coord operator =() \n";

x = ob2.x;

y = ob2.y;

return \*this;

}

class Quad : public Coord {

int quadrant;

public:

Quad() { x = 0; y = 0; quadrant = 0; }

Quad(int x, int y) : Coord(x, y) {

if (x >= 0 && y >= 0)

quadrant = 1;

else if (x < 0 && y >= 0)

quadrant = 2;

else if (x < 0 && y < 0)

quadrant = 3;

else

quadrant = 4;

}

void showq() {

cout << " Point in Quadrant: " << quadrant << '\n';

}

Quad operator =(Coord ob2);

};

Quad Quad::operator =(Coord ob2) {

cout << " Using Quad operator =() \n";

x = ob2.x;

y = ob2.y;

if (x >= 0 && y >= 0)

quadrant = 1;

else if (x < 0 && y >= 0)

quadrant = 2;

else if (x < 0 && y < 0)

quadrant = 3;

else

quadrant = 4;

return \*this;

}

int main() {

Quad object1(10, 10), object2(15, 3), object3;

int x, y;

object3 = object1 + object2;

object3.get\_xy(x, y);

object3.showq();

cout << " object1 + object2 X: " << x << ", Y: " << y << "\n\n";

object3 = object1 - object2;

object3.showq();

cout << " object1 - object2 X: " << x << ", Y: " << y << "\n\n";

object3 = object1;

object3. get\_xy (x, y);

object3. showq ();

cout << " object3 = object1 X: " << x << ", Y: " << y << "\n";

return 0;

}

/\*У програмі створюються об’єкти типу *Quad*, відбувається їх перевірка, присвоєння значення *quadrant*, але оскільки цей клас дочірний до *Coord*, то спочатку викликається конструктор *Coord() { x = 0; y = 0; },* потім *Quad() { x = 0; y = 0; quadrant = 0; },* потім знову *Coord*. Тоді відбувається певна операція +/-/= функціями класу Coord і повернення за правилами наслідування до дочірного класу *Quad*, де викликається функція *Quad operator =(Coord ob2){ …}*, відбувається перевірка, присвоєння значення *quadrant,* повернення переданого об’єкту та вихід з функції. Викликається *void showq() {…},* потім виведення значення *x* i *y* .\*/

**Результати роботи програми (скриншоти):**

