Лабораторна робота 6(2)

Варіант 18

1)

#include <iostream>

using namespace std;

class Base1 {

private:

int x;

public:

Base1(int x) : x(x) {}

int getX() const { return x; }

};

class Derived1 : public Base1 {

private:

int y;

public:

Derived1(int x, int y) : Base1(x), y(y) {}

int getY() const { return y; }

};

class Base2 {

private:

int x;

public:

Base2(int x) : x(x) {}

virtual ~Base2() {}

int getX() const { return x; }

};

class Derived2 : public Base2 {

private:

int y;

public:

Derived2(int x, int y) : Base2(x), y(y) {}

int getY() const { return y; }

};

int main() {

Derived1 obj1(5, 10);

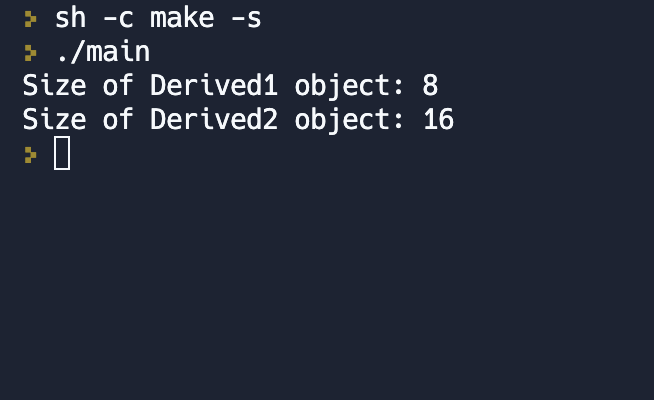
cout << "Size of Derived1 object: " << sizeof(obj1) << endl;

Derived2 obj2(5, 10);

cout << "Size of Derived2 object: " << sizeof(obj2) << endl;

return 0;

}



2)

#include <iostream>

#include <cmath>

using namespace std;

class Figure {

public:

virtual double surfaceArea() = 0;

};

class Parallelepiped : public Figure {

private:

double x, y, z;

public:

Parallelepiped(double a, double b, double c) {

x = a;

y = b;

z = c;

}

double surfaceArea() {

return 2 \* (x \* y + y \* z + z \* x);

}

};

class Tetrahedron : public Figure {

private:

double a;

public:

Tetrahedron(double edge) {

a = edge;

}

double surfaceArea() {

return pow(a, 2) \* sqrt(3);

}

};

class Sphere : public Figure {

private:

double r;

public:

Sphere(double radius) {

r = radius;

}

double surfaceArea() {

return 4 \* M\_PI \* pow(r, 2);

}

};

int main() {

Parallelepiped cube(2, 3, 4);

Tetrahedron pyramid(5);

Sphere ball(2.5);

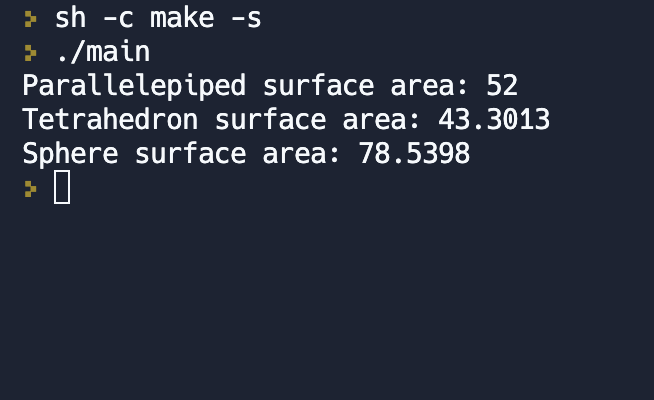
cout << "Parallelepiped surface area: " << cube.surfaceArea() << endl;

cout << "Tetrahedron surface area: " << pyramid.surfaceArea() << endl;

cout << "Sphere surface area: " << ball.surfaceArea() << endl;

return 0;

}



3)

#include <iostream>

#include <string>

using namespace std;

class WaterBody {

protected:

string name;

double area;

public:

WaterBody(string n = "", double a = 0) : name(n), area(a) {}

virtual ~WaterBody() {}

virtual void print() {

cout << "Water body name: " << name << ", Area: " << area << endl;

}

};

class Bay : public virtual WaterBody {

protected:

int numIslands;

public:

Bay(string n = "", double a = 0, int num = 0) : WaterBody(n, a), numIslands(num) {}

virtual ~Bay() {}

virtual void print() {

WaterBody::print();

cout << "Number of islands: " << numIslands << endl;

}

};

class Inlet : public virtual WaterBody {

protected:

double maxDepth;

public:

Inlet(string n = "", double a = 0, double d = 0) : WaterBody(n, a), maxDepth(d) {}

virtual ~Inlet() {}

virtual void print() {

WaterBody::print();

cout << "Maximum depth: " << maxDepth << endl;

}

};

class Sea : public Bay, public Inlet {

public:

Sea(string n = "", double a = 0, int num = 0, double d = 0) : WaterBody(n, a), Bay(n, a, num), Inlet(n, a, d) {}

virtual ~Sea() {}

virtual void print() {

WaterBody::print();

cout << "Sea information:" << endl;

Bay::print();

Inlet::print();

}

};

int main() {

WaterBody\* waterBodies[4];

waterBodies[0] = new WaterBody("Atlantic Ocean", 106400000);

waterBodies[1] = new Bay("Chesapeake Bay", 11444, 2000);

waterBodies[2] = new Inlet("Narragansett Bay", 147, 25);

waterBodies[3] = new Sea("Mediterranean Sea", 2500000, 3000, 5280);

for (int i = 0; i < 4; i++) {

waterBodies[i]->print();

cout << endl;

}

for (int i = 0; i < 4; i++) {

delete waterBodies[i];

}

return 0;

}

