Лабораторна робота 6(2)

1)

#include <iostream>

#include <string>

using namespace std;

class BaseException

{

protected:

string text;

public:

virtual string what() = 0;

};

class ArithmeticException :public BaseException

{

public:

ArithmeticException()

{

text = "Error. Arithmetic Exception.";

}

ArithmeticException(string \_text) { text = \_text; }

string what() { return text; }

};

class DivideByZero final :public ArithmeticException

{

public:

DivideByZero() :ArithmeticException()

{

text = "Divide by zero.";

}

DivideByZero(string \_text) :ArithmeticException(\_text)

{

}

string what()

{

return text;

}

};

class NegativeIndex final :public BaseException

{

public:

NegativeIndex(string \_text = "Error. Negative Index.") { text = \_text; }

string what() { return text; }

};

void DemoExceptions1()

{

int a[10];

int index;

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

a[i] = i \* i;

cout << "Input index: " << endl;

cin >> index;

if (index < 0)

throw NegativeIndex();

cout << "a[" << index << "] = " << a[index] << endl;

}

void DemoExceptions2()

{

int a, b, c;

cout << "a = "; cin >> a;

cout << "b = "; cin >> b;

if (b == 0)

throw DivideByZero("Divide by 0.");

cout << "a / b = " << (double)a / b << endl;

}

int main()

{

try

{

DemoExceptions1();

DemoExceptions2();

cout << "OK!" << endl;

}

catch (NegativeIndex e)

{

cout << e.what() << endl;

}

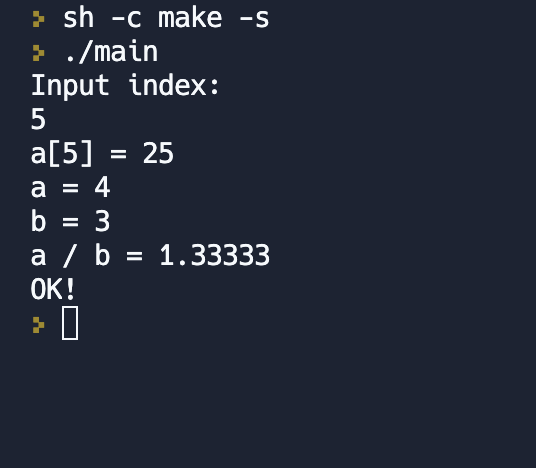
catch (DivideByZero e)

{

cout << e.what() << endl;

}

}



2)

#include <iostream>

#include <cmath>

using namespace std;

class Figure {

public:

virtual float Area() const = 0;

virtual ~Figure() {}

};

class Parallepiped : public Figure {

float x, y, z;

public:

Parallepiped() : x(0), y(0), z(0) {}

Parallepiped(float x1, float y1, float z1) : x(x1), y(y1), z(z1) {}

virtual float Area() const {

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

}

};

class Tetraid : public Figure {

float a;

public:

Tetraid() : a(0) {}

Tetraid(float a1) : a(a1) {}

virtual float Area() const {

return std::sqrt(3.) \* a;

}

};

class Piramida : public Figure {

float x, y, h;

public:

Piramida() : x(0), y(0), h(0) {}

Piramida(float x1, float y1, float h1) : x(x1), y(y1), h(h1) {}

virtual float Area() const {

return 0.;

}

};

class Kylia : public Figure {

float r;

public:

Kylia() : r(0) {}

Kylia(float r1) : r(r1) {}

virtual float Area() const {

return 4 \* std::acos(-1.) \* std::pow(r, 2.);

}

};

int main() {

Figure \* mas[4];

mas[0] = new Parallepiped(1, 3, 5);

mas[1] = new Piramida(2, 6, 10);

mas[2] = new Tetraid(24);

mas[3] = new Kylia(12);

cout << "1-Паралелепіпед" << endl << "2-Піраміда" << endl << "3-Тетраїдр"

<< endl << "4-Куля" << endl << endl;

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

cout << "Площа фігури, номер " << i + 1 << " = " << mas[i]->Area() << endl;

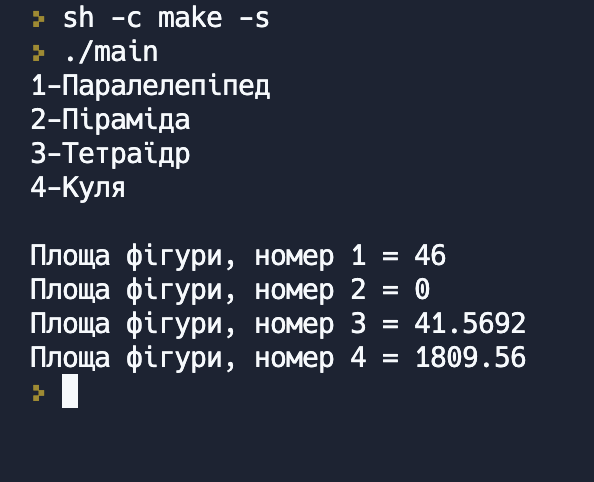
}

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

delete mas[i];

}

}



3)

#include <string>

#include <iostream>

using namespace std;

class Car {

protected:

string marka;

float power;

int numberOfWheels;

public:

Car() : marka("Tesla"), power(3.1f), numberOfWheels(4) {

}

Car(string m, float p, int nw) : marka(m), power(p), numberOfWheels(nw) {

}

string getMarka() { return marka; }

void setMarka(string m) { marka = m; }

float getPower() {

return power;

}

void setPower(float p) {

power = p;

}

int getNumberOfWheels() {

return numberOfWheels;

}

void setNumberOfWheels(int n) {

numberOfWheels = n;

}

string toString() {

string r= marka + "\t" + to\_string(power) + "\t" + to\_string(numberOfWheels) +

"\t";

return r;

}

};

class PassengerTransport {

protected:

int flightNumber;

int numberOfPassengerSeats;

public:

PassengerTransport() : flightNumber(112), numberOfPassengerSeats (5) {}

PassengerTransport(int f,int n) : flightNumber(f), numberOfPassengerSeats(n) {}

int getFlightNumber() { return flightNumber; }

void setFlightNumber(int f) { flightNumber =f; }

int getNumberOfPassengerSeats() { return numberOfPassengerSeats; }

void setnumberOfPassengerSeats(int n) { numberOfPassengerSeats=n; }

string toString() {

string r = to\_string(flightNumber) + "\t" + to\_string(numberOfPassengerSeats) +

"\t";

return r;

}

};

class AutoBus : public Car, public PassengerTransport

{

string busRoute;

public:

AutoBus() : busRoute("Kyiv-Rivne") {}

AutoBus(string m, float p, int nw, int f, int n, string bs)

: Car(m,p,nw), PassengerTransport(f,n), busRoute(bs) {}

string getbusRoute() {

return busRoute; }

void setbusRoute(string bs) { busRoute=bs; }

string toString() {

string r = Car::toString() + PassengerTransport::toString() + busRoute;

return r;

}

};

int main()

{

AutoBus def;

AutoBus lvCh("Chevrolet",2.8f,4,115,5,"Lviv-Odessa");

AutoBus\* pVnCn = new AutoBus();

pVnCn->setMarka("Toyota");

pVnCn->setPower(3.5f);

pVnCn->setNumberOfWheels(4);

pVnCn->setFlightNumber(111);

pVnCn->setnumberOfPassengerSeats(5);

pVnCn->setbusRoute("Chernihiv-Chernivci");

cout << def.toString() << endl;

cout << lvCh.toString() << endl;

cout << pVnCn->toString() << endl;

return 0;

}

