МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ»)

Факультет информационных технологий компьютерной безопасности

Кафедра автоматизированных вычислительных систем

**ОТЧЕТ**

По лабораторной работе №9 по дисциплине

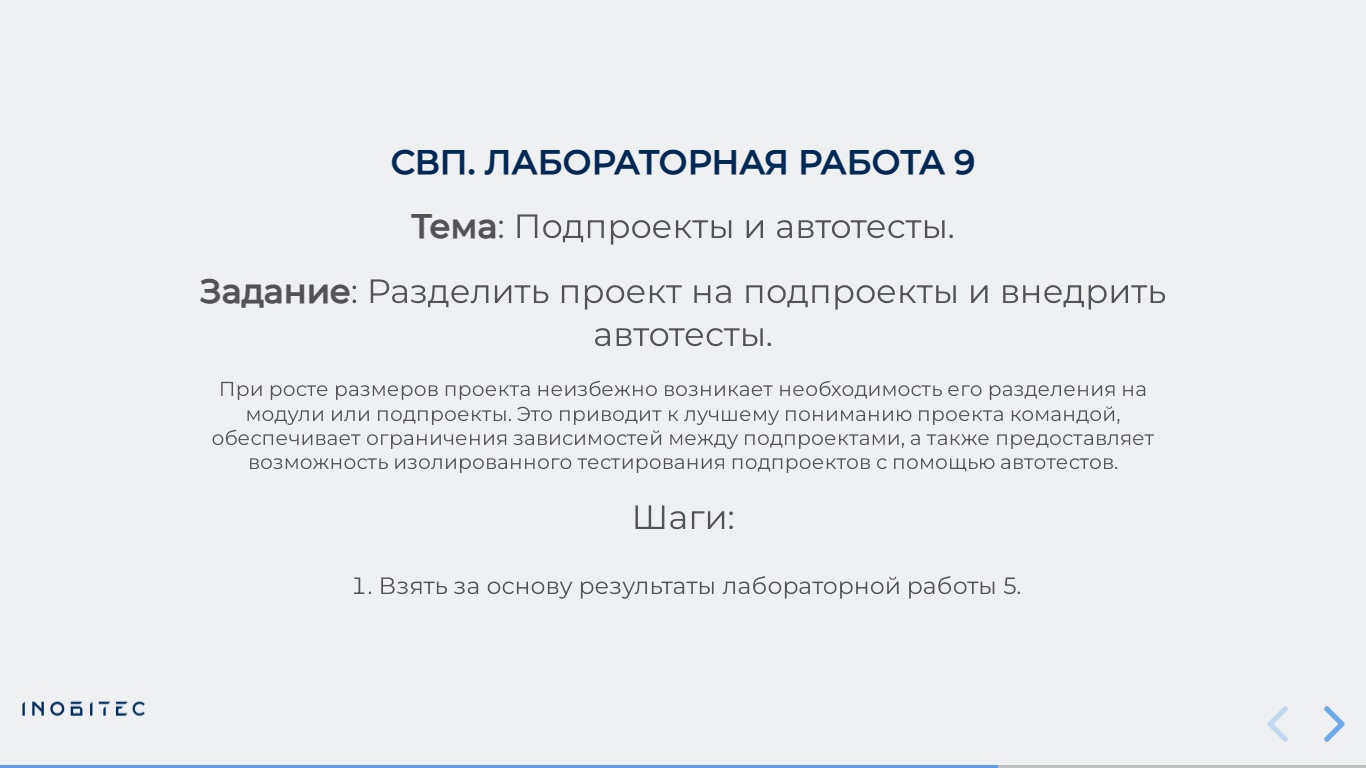
«Среды визуального программирования»

|  |  |  |
| --- | --- | --- |
| Выполнил |  | Сафарзода Ф.Ф |
|  |  | инициалы, фамилия |

|  |  |  |
| --- | --- | --- |
| Руководитель |  | Сокольников В.В. |
|  |  | инициалы, фамилия |

«15 » октября 2023 г.

Воронеж 2023



**Листинг svp\_lab9.pro:**

TEMPLATE = subdirs

SUBDIRS += \

objects \

app \

test

CONFIG += ordered

Листинг app.pro:

TEMPLATE = app

CONFIG += console c++11

CONFIG -= app\_bundle

CONFIG -= qt

CONFIG(debug, debug|release) {

LIBS += -L../objects/debug -lobjects

}

CONFIG(release, debug|release) {

LIBS += -L../objects/release -lobjects

}

SOURCES += \

main.cpp

**Листинг main.cpp:**

#include "../objects/Objects.h"

int main(int argc, char\* argv[])

{

if (argc < 2)

{

cout << "Usage: program\_name input\_file" << endl;

return 1;

}

string inputFileName = argv[1];

ifstream inputFile(inputFileName);

if (!inputFile)

{

cout << "Failed to open input file." << endl;

return 1;

}

Bicycle bicycle;

Sedan sedan;

Suv suv;

Bus bus;

Route route;

RoutePoint point1 = {0, 0, "Point 1"};

RoutePoint point2 = {10, 10, "Point 2"};

RoutePoint point3 = {20, 20, "Point 3"};

route.addPoint(point1);

route.addPoint(point2);

route.addPoint(point3);

inputFile.close();

cout << "Bicycle:" << endl;

route.run(&bicycle);

bicycle.printStatus();

cout << "Sedan:" << endl;

route.run(&sedan);

sedan.printStatus();

cout << "Suv:" << endl;

route.run(&suv);

suv.printStatus();

cout << "Bus:" << endl;

route.run(&bus);

bus.printStatus();

return 0;

}

**Листинг objects.pro:**

CONFIG -= qt

TEMPLATE = lib

CONFIG += staticlib

CONFIG += c++11

DEFINES += QT\_DISABLE\_DEPRECATED\_BEFORE=0x060000

SOURCES += \

Objects.cpp

HEADERS += \

Objects.h

# Default rules for deployment.

unix {

target.path = $$[QT\_INSTALL\_PLUGINS]/generic

}

!isEmpty(target.path): INSTALLS += target

**Листинг Objects.h:**

#include <iostream>

#include <string>

#include <vector>

#include <fstream>

#include <cmath>

using namespace std;

class IMachine

{

public:

virtual bool drive(const int driven) = 0;

virtual void reBenzin() = 0;

virtual void printStatus() const = 0;

virtual int getMilage() const = 0;

};

struct RoutePoint

{

int xKm, yKm;

string name;

RoutePoint(const int xKm, const int yKm, const string& name) : xKm(xKm), yKm(yKm), name(name) {}

};

class AbstractCar : public IMachine

{

protected:

AbstractCar();

public:

bool drive(const int driven) override;

void reBenzin() override;

void printStatus() const override;

int getMilage() const override;

double getBenzinLevel() const;

protected:

double BenzinTrata;

double Benzin;

double Benzmax;

int probeg;

string m\_name;

};

class Sedan : public AbstractCar

{

public:

Sedan();

};

class Suv : public AbstractCar

{

public:

Suv();

};

class Bus : public AbstractCar

{

public:

Bus();

};

class Bicycle : public IMachine

{

public:

Bicycle();

bool drive(const int driven) override;

void reBenzin() override;

void printStatus() const override;

int getMilage() const override;

private:

int probeg;

string m\_name;

};

class Route

{

public:

Route() {}

void addPoint(const RoutePoint& point);

void run(IMachine\* vehile);

private:

vector<RoutePoint> m\_path;

};

**Листинг Objects.cpp:**

#include "Objects.h"

AbstractCar::AbstractCar() : BenzinTrata(0.0), Benzin(0.0), Benzmax(0.0), probeg(0)

{}

void AbstractCar::void reBenzin()

{

Benzin = Benzmax;

}

bool AbstractCar::drive(int driven)

{

if (Benzin < driven \* BenzinTrata)

return false;

probeg += driven;

Benzin -= driven \* BenzinTrata;

return true;

}

void AbstractCar::printStatus()

{

cout << "Benzin= " << Benzin << endl;

cout << "Drive= " << probeg << endl;

}

int AbstractCar::getProbeg() const

{

return probeg;

}

double AbstractCar::getBenzinLevel() const

{

return Benzin;

}

Sedan::Sedan(){

BenzinTrata = 7.0;

Benzmax = 50;

m\_name = "Sedan";

}

Suv::Suv(){

BenzinTrata= 11.0;

Benzmax = 80.0;

m\_name = "Suv";

}

Bus::Bus(){

BenzinTrata = 25.0;

Benzmax = 120.0;

m\_name = "Bus";

}

Bicycle::Bicycle() : probeg(0), m\_name("Bicycle") {}

bool Bicycle::drive(const int driven)

{

probeg += driven;

return true;

}

void Bicycle::reBenzin() {}

void Bicycle::printStatus() const

{

cout << m\_name << " probeg pedalnogo: " << probeg << endl;

}

int Bicycle::getMilage() const

{

return probeg;

}

void Route::addPoint(const RoutePoint& point)

{

m\_path.push\_back(point);

}

void Route::run(IMachine\* machine)

{

if (m\_path.size() < 2)

{

cout << " Malo tochek v marshrute! " << endl;

return;

}

machine->printStatus();

for (size\_t i = 1, t = m\_path.size(); i < t; ++i)

{

const RoutePoint& prevPoint = m\_path[i - 1];

const RoutePoint& currPoint = m\_path[i];

const int xDistance = currPoint.xKm - prevPoint.xKm;

const int yDistance = currPoint.yKm - prevPoint.yKm;

const int distance = (int)sqrt(xDistance \* xDistance + yDistance \* yDistance);

for (int j = 0; j < 2; ++j)

{

if (!machine->drive(distance))

{

if (j == 0)

{

machine->reBenzin();

continue;

}

else

{

cout << "Unable to drive the rout" << endl;

return;

}

}

machine->printStatus();

break;

}

}

}

**Листинг test.pro:**

QT += testlib

QT -= gui

CONFIG += qt console warn\_on depend\_includepath testcase

CONFIG -= app\_bundle

TEMPLATE = app

CONFIG(debug, debug|release) {

LIBS += -L../objects/debug -lobjects

}

CONFIG(release, debug|release) {

LIBS += -L../objects/release -lobjects

}

SOURCES += tst\_test.cpp

**Листинг tst\_test.cpp:**

#include <QtTest>

#include "../objects/Objects.h"

// add necessary includes here

class Test : public QObject

{

Q\_OBJECT

public:

Test();

~Test();

private slots:

void test\_sedans();

};

Test::Test(){}

Test::~Test(){}

void Test::test\_sedans()

{

Sedan sedan;

QVERIFY(sedan.drive(0)); // Пытаемся проехать 0 км. Проверяем, что метод вернул true

QCOMPARE(sedan.getMilage(), 0); // Проверяем, что пробег остался на 0 км

QVERIFY(!sedan.drive(100)); // Пытаемся проехать 100 км. Проверяем, что метод вернул false, т.к. седан не заправлен

QCOMPARE(sedan.getBenzinLevel(), 0.0); // Проверяем, что бак пуст

sedan.refuel();

QCOMPARE(sedan.getBenzinLevel(), 50.0); // Проверяем, что бак заполнен

QVERIFY(sedan.drive(100)); // Пытаемся проехать 100 км. Проверяем, что метод вернул true

QCOMPARE(sedan.getMilage(), 100); // Проверяем пробег

QCOMPARE(sedan.getBenzinLevel(), 43.0); // Проверяем остаток в баке

QVERIFY(sedan.drive(150)); // Пытаемся проехать 150 км. Проверяем, что метод вернул true

QCOMPARE(sedan.getMilage(), 250); // Проверяем пробег

QCOMPARE(sedan.getFBenzinLevel(), 32.5); // Проверяем остаток в баке

QVERIFY(!sedan.drive(500)); // Пытаемся проехать 500 км. Проверяем, что метод вернул false

QCOMPARE(sedan.getMilage(), 250); // Проверяем пробег

QCOMPARE(sedan.getBenzinLevel(), 32.5); // Проверяем остаток в баке

sedan.refuel();

QCOMPARE(sedan.getBenzinLevel(), 50.0); // Проверяем, что бак заполнен

QVERIFY(sedan.drive(500)); // Пытаемся проехать 500 км. Проверяем, что метод вернул true

QCOMPARE(sedan.getMilage(), 750); // Проверяем пробег

QCOMPARE(sedan.getBenzinLevel(), 15.0); // Проверяем остаток в баке

}

QTEST\_APPLESS\_MAIN(Test)

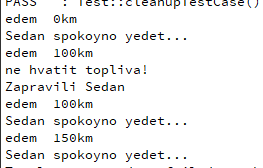
#include "tst\_test.moc"

Рис.1 — Результат тестирования программы