Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«Воронежский государственный технический университет»

(ФГБОУ ВО «ВГТУ», ВГТУ)

Отчет по предмету Среды визуального программирования

Лабораторная 4.

Обучающийся Копылов Александр Александрович

(*ФИО обучающегося*)

Группа бИВТ-222

Наименование предприятия ВГТУ

Обучающийся \_\_\_\_\_\_\_\_\_\_ А.А. Копылов

*(подпись)*

Руководитель по практической подготовке \_\_\_\_\_\_\_\_\_\_ А.С. Троценко

*(подпись)*

Воронеж 2023

Тема: «Контейнеры и вложенные циклы в С++.

Расширение функциональности.»

Задание:

Моделирование процесса движения транспортного средства по заданному маршруту. Консольное приложение.

Листинг.

#include <iostream>

#include <iomanip>

#include <vector>

#include <string>

#include <cmath>

using namespace std;

// Абстрактный класс для транспорта

class IVehicle

{

public:

virtual bool drive(int kilometers) = 0;

virtual void refuel() = 0;

virtual void printStatus() = 0;

virtual ~IVehicle() = default;

};

// Абстрактный класс для автомобиля

class AbstractCar : public IVehicle

{

private:

int zapastopliva;

int rashodna100km;

int ostatokLtr;

int probeg;

int maxBak;

int maxprobeg;

int vsegoproehal;

bool hasDriven;

public:

AbstractCar(int maxFuelCapacity, int maxRange, int fuelConsumption)

: zapastopliva(maxFuelCapacity), maxBak(maxFuelCapacity), maxprobeg(maxRange),

rashodna100km(fuelConsumption), probeg(0), ostatokLtr(maxFuelCapacity),

vsegoproehal(0), hasDriven(false) {}

bool drive(int kilometers) override

{

if (ostatokLtr >= kilometers / rashodna100km)

{

probeg += kilometers;

ostatokLtr -= kilometers / rashodna100km;

hasDriven = true;

cout << "Вы проехали: " << kilometers << " км." << endl;

return true;

}

else

{

cout << "Не хватает топлива для поездки!" << endl;

return false;

}

}

void refuel() override

{

int neededFuel = maxBak - ostatokLtr;

ostatokLtr += neededFuel;

cout << "Вы заправились!" << endl;

}

void printStatus() override

{

if (hasDriven)

{

vsegoproehal++;

cout << setw(10) << vsegoproehal << " Поездка" << endl;

cout << "Пробег: " << probeg << " км" << endl;

cout << "Оставшееся топливо: " << ostatokLtr << " литров" << endl;

cout << "Запас хода: " << ostatokLtr \* rashodna100km << " км" << endl;

cout << endl;

if (vsegoproehal == 4)

{

cout << endl;

}

}

}

~AbstractCar() {}

};

// Класс для велосипеда

class Bicycle : public IVehicle

{

private:

int mileage;

public:

Bicycle() : mileage(0) {}

bool drive(int kilometers) override

{

mileage += kilometers;

return true;

}

void refuel() override {}

void printStatus() override

{

cout << "Вы проехали " << mileage << " км на велосипеде." << endl;

}

};

// Класс для седана

class Sedan : public AbstractCar

{

public:

Sedan() : AbstractCar(43, 280, 7) {}

};

// Класс для внедорожника

class Suv : public AbstractCar

{

public:

Suv() : AbstractCar(70, 280, 25) {}

};

// Класс для автобуса

class Bus : public AbstractCar

{

public:

Bus() : AbstractCar(120, 600, 20) {}

};

// Класс для точки маршрута

class RoutePoint

{

public:

int xKm;

int yKm;

string nameRoutePoint;

IVehicle\* toVehicle() const

{

// TODO: Непонятный код. Что вы хотели сказать?

RoutePoint point;

IVehicle\* vehicle = point.toVehicle();

}

RoutePoint() : xKm(0), yKm(0), nameRoutePoint("Default") {}

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

~RoutePoint() = default;

};

// Класс для маршрута

class Route : public RoutePoint

{

private:

vector<RoutePoint> routePoints;

public:

Route() = default;

void addPoint(const RoutePoint& point)

{

routePoints.push\_back(point);

}

void run(IVehicle\* vehicle)

{

for (const auto& point : routePoints)

{

if (xKm >= 0 && yKm >= 0)

{

cout << "Текущая точка маршрута: " << "(" << point.xKm << "," << point.yKm << ") " << point.nameRoutePoint << std::endl;

double hypotenuse = sqrt(pow(point.xKm, 2) + pow(point.yKm, 2));

vehicle->drive(hypotenuse);

vehicle->refuel();

vehicle->printStatus();

}

}

}

~Route() = default;

};

int main()

{

setlocale(LC\_ALL, "Russian");

Route routes[2];

routes[0].addPoint(RoutePoint(0, 0, "Point1"));

routes[0].addPoint(RoutePoint(100, 0, "Point2"));

routes[0].addPoint(RoutePoint(200, 100, "Point3"));

routes[0].addPoint(RoutePoint(300, 300, "Point4"));

routes[0].addPoint(RoutePoint(100, 400, "Point5"));

routes[1].addPoint(RoutePoint(0, 0, "Moscow"));

routes[1].addPoint(RoutePoint(0, 540, "Voronezh"));

routes[1].addPoint(RoutePoint(420, 540, "Saratov"));

IVehicle\* vehicles[4] = { new Sedan, new Suv, new Bus, new Bicycle };

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

{

IVehicle\* vehicle = vehicles[i];

cout << "Транспортное средство: ";

if (dynamic\_cast<Sedan\*>(vehicle) != nullptr) cout << "Седан" << endl;

else if (dynamic\_cast<Suv\*>(vehicle) != nullptr) cout << "Внедорожник" << endl;

else if (dynamic\_cast<Bus\*>(vehicle) != nullptr) cout << "Автобус" << endl;

else if (dynamic\_cast<Bicycle\*>(vehicle) != nullptr) cout << "Велосипед" << endl;

routes[0].run(vehicle);

routes[1].run(vehicle);

delete vehicle;

cout << endl;

}

return 0;

}



