**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

Лабораторная работа № 7

по ОБЪЕКТНО-ОРИЕНТИРОВАННОМУ ПРОГРАММИРОВАНИЮ

«Обработка собственных событий»

Студент \_\_\_\_\_\_\_\_\_\_\_ Целищев А.Е.

Группа ПМ-21-2

Руководитель

Доцент \_\_\_\_\_\_\_\_\_\_\_ Кургасов В.В.

Липецк 2022г.

Оглавление

Цель работы 2

Задание кафедры 2

Код программы 3

Пример работы программы 11

Вывод 13

Контрольные вопросы...........................................................................................14

**Цель работы:**

Научиться создавать и использовать собственные события для возможности отслеживания другими классами наступления определённых условий в текущем классе.

**Задание кафедры:**

Реализовать на языке С++ программу, требования для которой представлены в таблице, с использованием механизма событий.

Задачи для реализации механизма событий

|  |  |  |
| --- | --- | --- |
| Вариант | Описание задачи | События |
| 11 | Программа-редактор гаражей и транспортных средств в нём. Имеются классы: гараж (более чем на одно место), транспортное средство. Программа должна предоставлять возможность редактировать списки гаражей и транспортных средств в них. Необходим класс с общим перечнем транспортных средств (с указанием расположения), в который транспортные средства будут попадать по событиям. Необходим класс для сохранения сообщений в файл о сработавших событиях. | 1. Создание транспортного средства; 2. Удаление транспортного средства; 3. Редактирование транспортного средства. |

***Код программы***

#include "pch.h"

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

using namespace System;

using namespace std;

// Объявление с использованием делегата метода для обработки события

delegate void MethodContainer(int count);

delegate void newVehicle(string su);

delegate void deleteVehicle(string su);

delegate void editedVehicle(string su);

ref class Writer

{

public:

event MethodContainer^ onCount; // объявление события onCount c типом делегата MethodContainer

event newVehicle^ nVehicle; // объявление события nVehicle(новое трансп. ср-во в некотором гараже) c типом делегата newVehicle

event deleteVehicle^ delVehicle;

event editedVehicle^ edVehicle;

void newVehicle(string su)

{

nVehicle(su);

}

void deleteVehicle(string su)

{

delVehicle(su);

}

void editedVehicle(string su)

{

edVehicle(su);

}

};

// классы ClassEvent реагируeт на событие класса Writer записью строки в консоль.

ref class ClassEvent

{

public:

void Message(string su)

{

cout << "New vehicle is added " << su << endl;

}

};

class Vehicle // Создаём класс транспортного средства

{

private:

string type, name, color; // Вид транспорта, название, цвет;

int garage; // Гараж, в который поместить транспорт

public:

Vehicle(string sType, string sName, string sColor, int iGarage)

{

type = sType;

name = sName;

color = sColor;

garage = iGarage;

}

// Вывод информации об объекте транспорта

void showVehicle()

{

cout << "Type: " << type << endl;

cout << "Name: " << name << endl;

cout << "Color: " << color << endl;

cout << "Location: Garage " << garage << endl;

}

// сеттеры и геттеры для взаимодействия с некоторыми приватными полями класса

void setName(string sName) { name = sName; }

void setColor(string sColor) { color = sColor; }

void setGarage(int iGarage) { garage = iGarage; }

string getType() { return type; }

string getName() { return name; }

string getColor() { return color; }

int getGarage() { return garage; }

};

// Класс гараж для хранения экземляров класса транспорт

class Garage

{

private:

int number;

vector <Vehicle> garage;

public:

// При создании присваиваем гаражу номер

Garage(int iNumber) { number = iNumber; }

// Геттер для получения номера гаража

int getNumber() { return number; }

// Добавление транспорта в гараж

void addVehicle(Vehicle newVehicle) { garage.push\_back(newVehicle); }

// Удаление транспорта по названию и цвету

void deleteVehicle(string sName, string sColor)

{

vector<Vehicle> nGarage;

for (auto &v : garage)

{

if (v.getName() != sName || v.getColor() != sColor)

nGarage.push\_back(v);

}

garage = nGarage;

}

// Редактирование информации о выбранном по названию и цвету транспорте

// Возвращает отредактированный объект - это нужно для записи в логи в процессе работы с классом

Vehicle editVehicle(string sName, string sColor)

{

vector<Vehicle> nGarage;

for (auto &v : garage)

{

if (v.getName() == sName && v.getColor() == sColor)

{

Vehicle nVehicle(v.getType(), sName, sColor, v.getGarage());

cout << "What information do you want to edit?\n1 - name\n2 - color\n3 - garage" << endl;

int ans; cin >> ans;

if (ans == 1)

{

cout << "Enter new vehicle's name: " << endl;

string newName; cin >> newName;

nVehicle.setName(newName);

}

else if (ans == 2)

{

cout << "Enter new vehicle's color: " << endl;

string newColor; cin >> newColor;

nVehicle.setColor(newColor);

}

else if (ans == 3)

{

cout << "Enter new garage's number to replace trasfer the vehicle:" << endl;

int newGarage; cin >> newGarage;

nVehicle.setGarage(newGarage);

}

nGarage.push\_back(nVehicle);

return nVehicle;

}

else { nGarage.push\_back(v); }

}

garage = nGarage;

}

// Вывод информации о транспортных средствах в гараже

void showGarage()

{

cout << "Garage number " << number << endl;

for (auto v : garage) {v.showVehicle();}

}

};

void event(string str)

{

Writer^ writer = gcnew Writer();

ClassEvent^ Event = gcnew ClassEvent();

writer->newVehicle(str);

writer->deleteVehicle(str);

writer->editedVehicle(str);

}

int main()

{

vector<Garage> garages; // Контейнер для хранения гаражений

ofstream file("logs.txt");

// Цикл работает до тех пор, пока пользователь не выйдет из программы (пока ans != 5)

while (true)

{

cout << "\nChoose option:" << endl;

cout << "1) Add vehicle" << endl;

cout << "2) Edit vehicle" << endl;

cout << "3) Delete Vehicle" << endl;

cout << "4) Show garages list" << endl;

cout << "5) Exit program\n" << endl;

int ans; cin >> ans;

if (ans == 1) // Добавление транспорта

{

cout << "Enter new vehicle's type, name, color and number of garage for it:" << endl;

string sType, sName, sColor; int iGarage;

cin >> sType >> sName >> sColor >> iGarage;

Vehicle vehicle(sType, sName, sColor, iGarage);

bool found = false;

for (auto &v : garages)

{

if (v.getNumber() == iGarage)

{

found = true;

v.addVehicle(vehicle);

}

}

if (!found)

{

Garage garage(iGarage);

garage.addVehicle(vehicle);

garages.push\_back(garage);

}

// Передаём название транспорта с помощью сеттера

event(vehicle.getName());

// Запись в файл

file << vehicle.getName() << " was added into garage " << vehicle.getGarage() << "\n";

}

else if (ans == 2)

{

cout << "Enter vehicle's garage, name and color to edit:" << endl;

int iGarage; cin >> iGarage;

string sName, sColor; cin >> sName >> sColor;

for (auto g : garages)

{

if (g.getNumber() == iGarage)

{

Vehicle newV = g.editVehicle(sName, sColor);

event(newV.getName());

file << sName << sColor << " in garage " << iGarage << " was edited and became ";

file << newV.getName() << " " << newV.getColor() << " in garage " << newV.getGarage() << endl;

break;

}

else { cout << "The vehicle is not found!" << endl; }

}

}

else if (ans == 3) // Удаление транспорта

{

cout << "Enter vehicle's garage, name and color to delete:" << endl;

int iGarage; cin >> iGarage;

string sName, sColor; cin >> sName >> sColor;

for (auto g : garages)

{

if (g.getNumber() == iGarage) { g.deleteVehicle(sName, sColor); }

}

event(sName);

file << sName << " in garage " << iGarage << " was deleted" << endl;

}

else if (ans == 4) // Вывод транспортных средств

{

for (auto g : garages)

{

g.showGarage();

}

}

else if (ans == 5) { break; }

else { cout << "Unknown input!" << endl; }

}

file.close();

return 0;

}

***Пример работы программы***

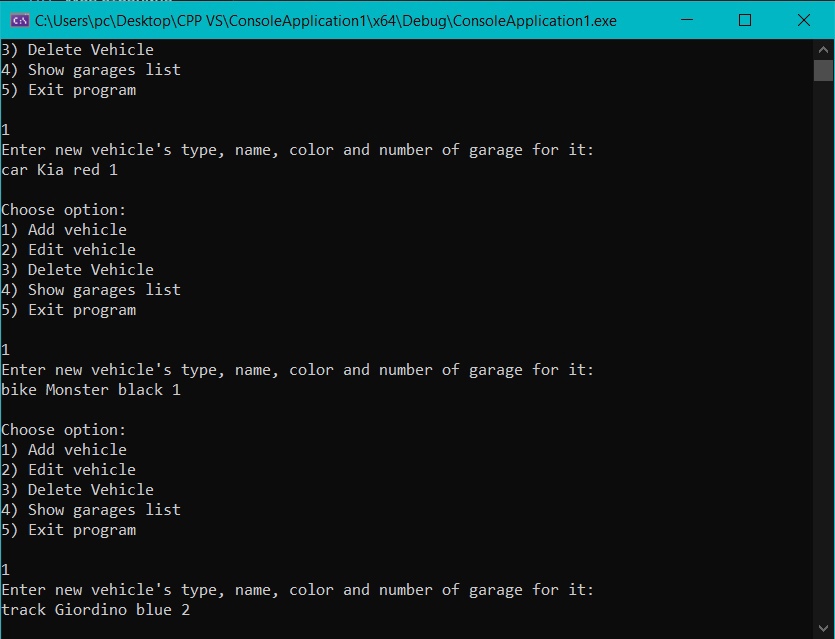


Рисунок 1. Добавление транспортных средств в гаражи

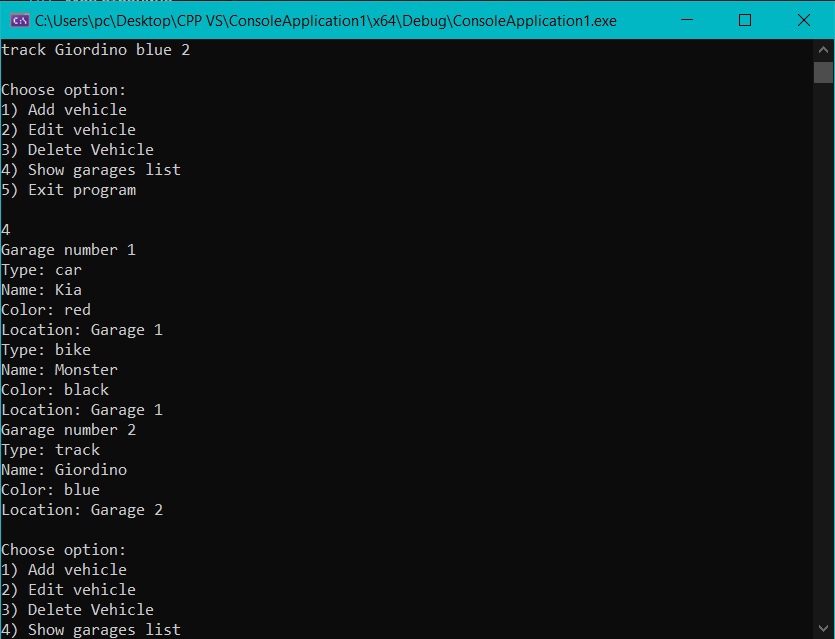


Рисунок 2. Вывод имеющихся транспортных средств в гаражах

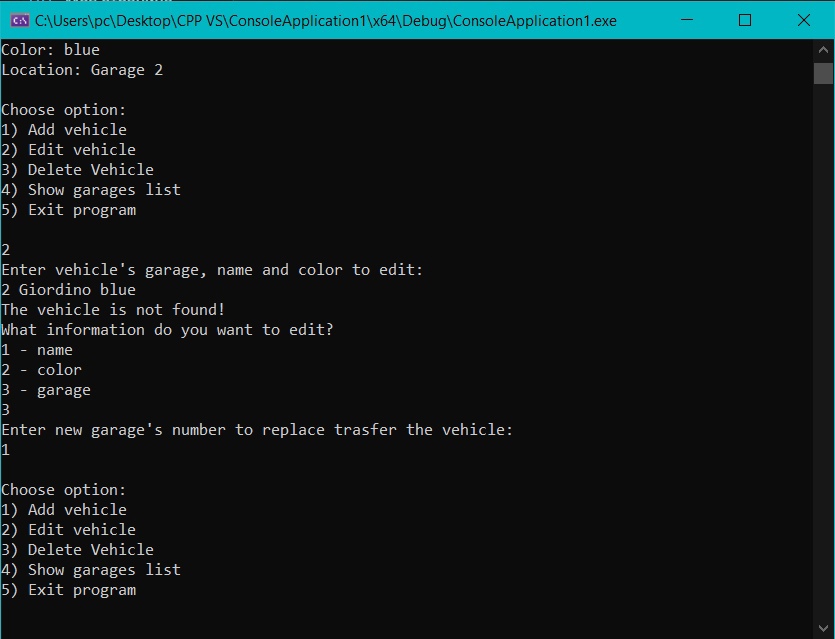


Рисунок 3. Редактирование информации о транспортных средствах



Рисунок 4. Результат логирования событий в текстовый файл logs.txt

***Вывод:***

На языке С++ реализована программа-редактор гаражей и транспортных средств в них с возможностью редактирования списков гаражей и транспортных средств в них. Также реализована обработка событий добавления, удаления, редактирования транспортных средств, а происходящие события записываются в отдельный текстовый файл для отслеживания работы программы (реализовано логирование событий).

***Контрольные вопросы***

1. **Что такое делегат?**

Делегат - это указатель на статическую функцию или нестатический метод какого-либо класса. Сигнатура, указанная в описании делегата, должна совпадать с сигнатурой метода, на который указывает делегат.

1. **Что такое событие? Как его можно создать?**

Событие - некоторое состояние объекта класса, которому присвоено имя и которое может быть обработано некоторым кодом, который исполняется только при возникновении данного события. В контексте языка С++ событие - это объект, отдельные поля которого характеризуют некоторые свойства передаваемой информации. Создать событие можно с использованием следующего синтаксиса:

event SOME\_DELEGATE ^ EVENT\_NAME;

Где SOME\_DELEGATE - делегат, с которым связывается событие.

1. **Как передать в событие один или несколько параметров?**

При определении делегата, соответствующего событию, передать ему в круглых скобках после имени нужные параметры, как это делалось бы в случае с обычными функциями или методами.

1. **Что такое обработчик события?**

Обработчик события - метод с сигнатурой (возвращаемым типом, аргументами, соглашением о вызовах), которая соответствует событию, которое он будет обрабатывать, т.е. исполнять некоторый код в случае наступления отслеживаемого события.

1. **Как отслеживать наступление события?**

Для этого нужно реализовать метод, соединяющий сигнал со слотом. Например, при нажатии кнопки мыши генерируется сигнал щелчка кнопки, и срабатывает указатель на метод-обработчик события. Далее программа работает в обычном режиме согласно её логике.