Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Петрозаводский государственный университет»

Физико-технический институт

Кафедра информационно-измерительных систем и физической электроники

Отчет по командному заданиюДисциплина: «Технология программирования»

Выполнили студенты: 3 курса  
физико-технического института,  
группы 21318

Петров Андрей Александрович  
 Степаненков Антон Сергеевич  
Сенкевич Ксения Сергеевна

Научный руководитель:   
кандидат физико-математических наук,  
Бульба Артем Владимирович

Цель работы:

Разработать программу для заказчика (в нашем случае заказчиком является сотрудник МВД), которая могла бы: выполнять ввод новых данных, производить редактирование уже имеющихся данных и выводить отсортированные списки(таблицы) с необходимой информацией.

Кратко о программной реализации:

Среда разработки: Microsoft Visual Studio 13.

Язык: C++.

Перечисление всех созданных заголовочников и единиц компиляции:

Заголовочинки:

Driver.h -Содержит информацию о конкретном водителе

DriverInputScreen.h – Содержит ряд переменных и функцию для заполнения нового водителя

DriverList.h -Содержит список всех водителей

GlobalMethods.h – Ряд глобальных методов

InterfaceMenu.h – Содержит в себе главное меню. С помощью него совершается переход по другим параметрам (таблицам)

Исходные файлы:

GlobalMethods.cpp – Содержит в себе реализацию глобальных методов

Driver.cpp – Содержит в себе методы класса Driver

DriverInputScreen.cpp – Содержит в себе методы добавления данных о новом водителе

DriverList.cpp – Содержит в себе данные о всех водителях и так же методы для редактирования, поиска нарушителей и погибших

IntefraceMenu.cpp – Реализация методов главного меню, с помощью которых происходит переход по другим параметрам (таблицам)

OurProject.cpp – Содержит в себе главную функцию main, с помощью которой мы запускаем нашу программу

# Краткое словесное описание сюжета

К нам поступил заказ от местного отделения МВД по делам автотранспортных нарушений. Им требовалось, чтобы мы написали программу, с помощью которой они смогут перенести данные из бумажных архивов и внести новые в электронный вид быстро и удобно. В базе должны хранится следующие данные: ФИО, адрес, контактный номер телефона, количество нарушений, номер, марка автомобиля и статус жизни владельца. Из этого следует, что им так же требовалось, чтобы система могла быстро выдать список водителей с нарушениями и усопших водителей.

# Описание имеющихся у заказчика материалов (данных), с которыми ему приходится работать

Таблица 1. Общий вид основной таблицы

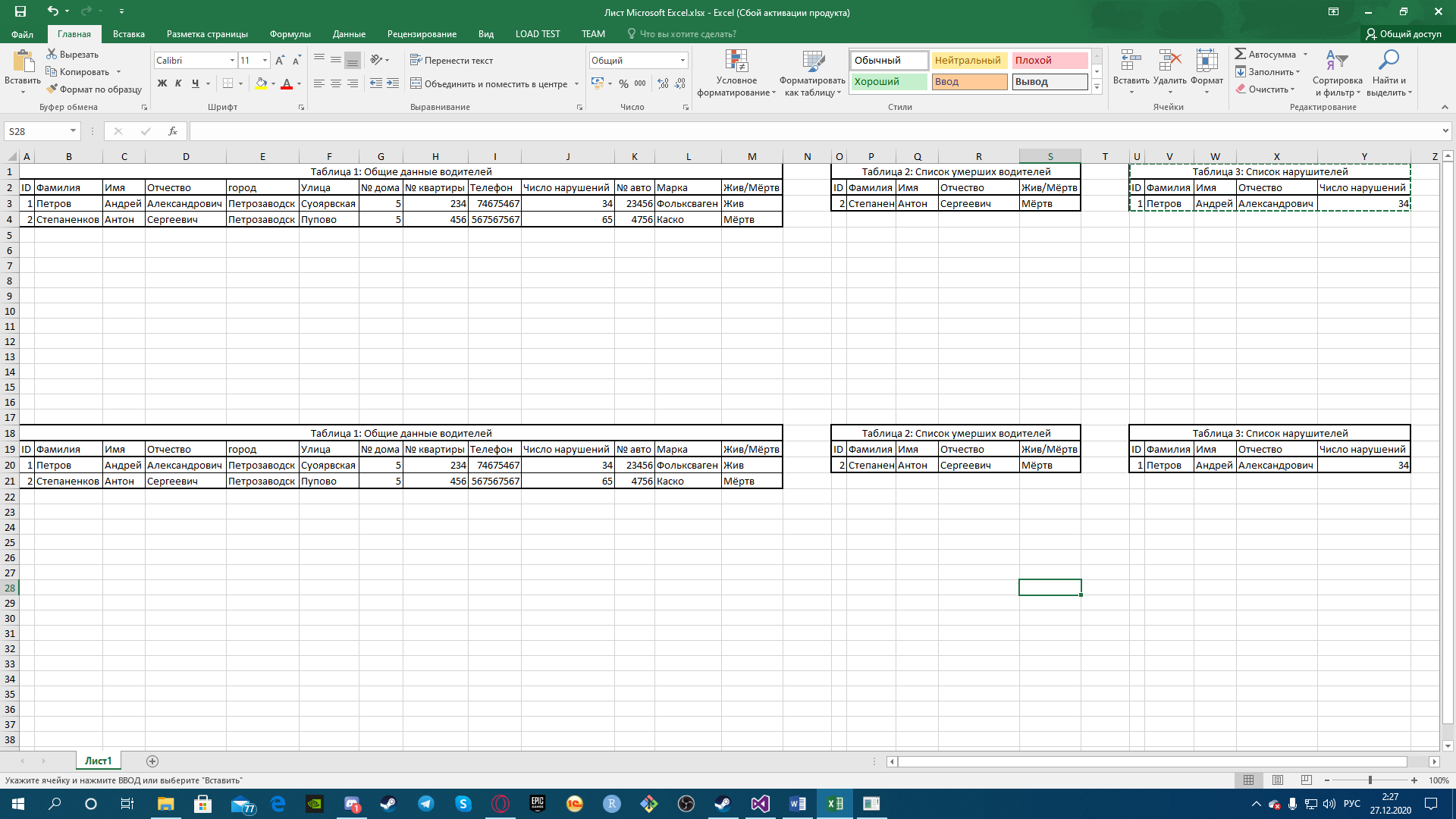


Таблица 2. Общий вид таблицы усопших водителей

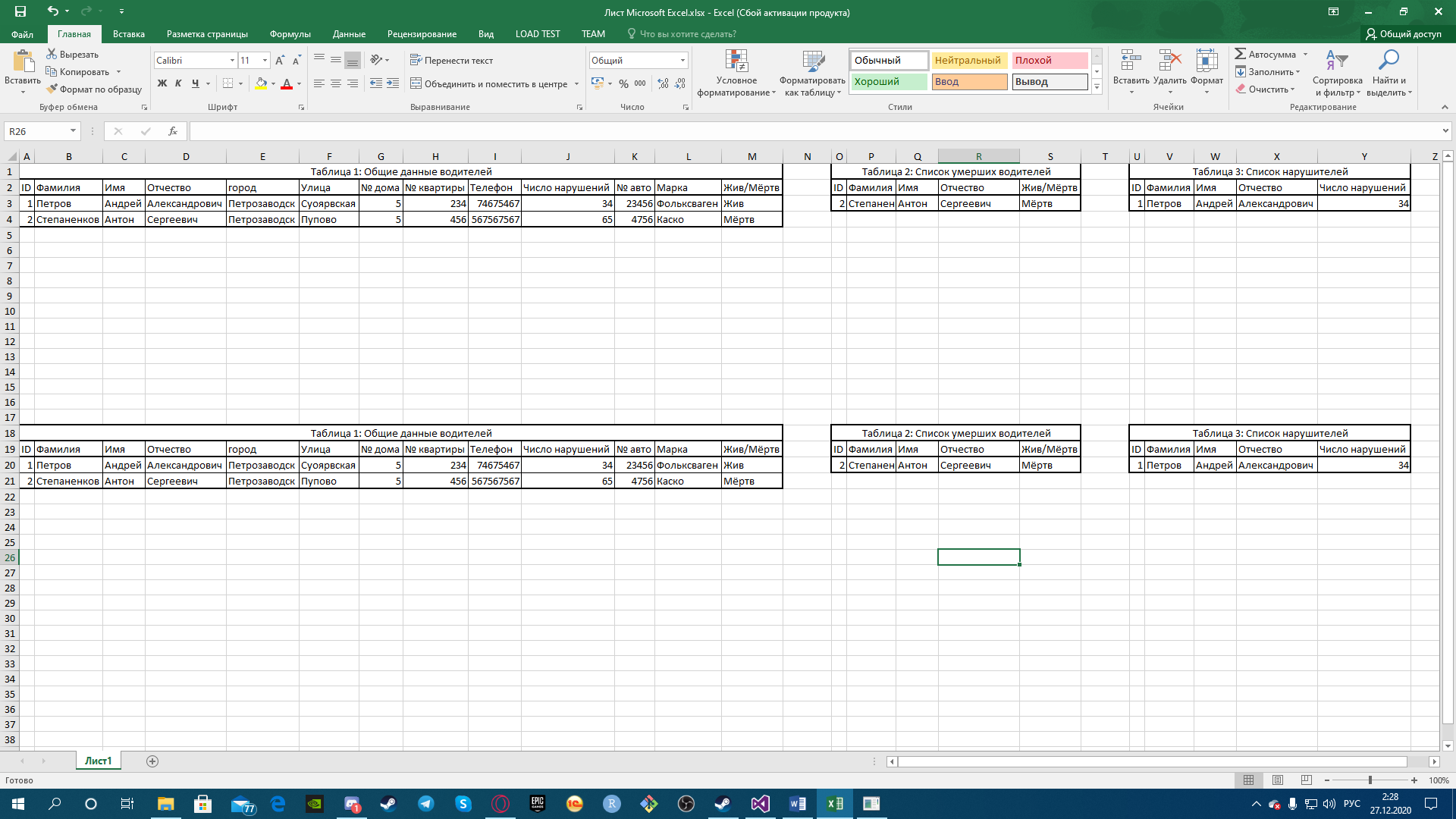
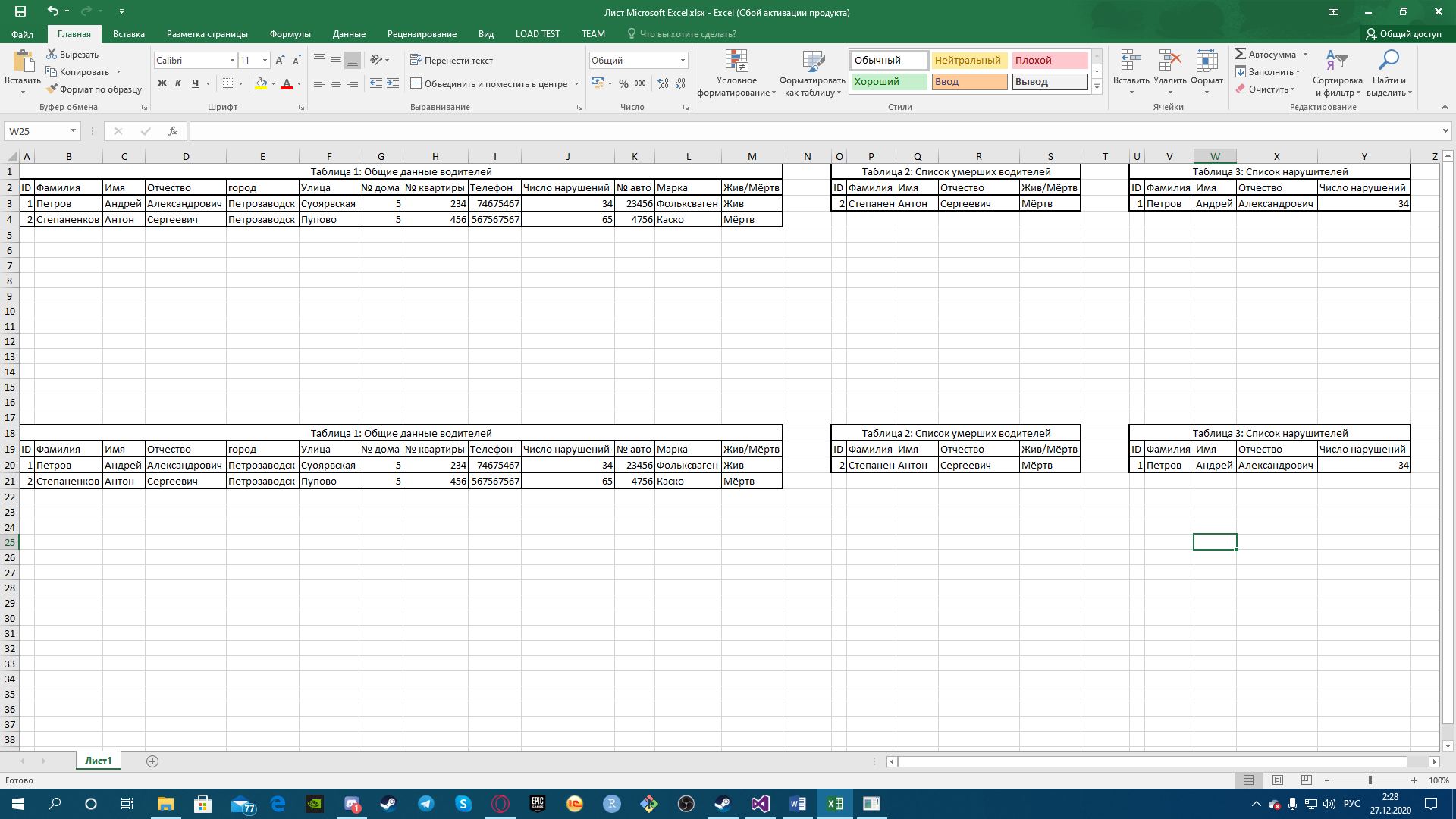


Таблица 3. Общий вид таблицы нарушителей



# Необходимо перечислить действующих субъектов (actors) и их функциональные обязанности

Наш заказчик - сотрудник МВД. В его полномочия входят: добавление, редактирование и просмотр данных.

# Представить список вариантов использования, которые могут возникнуть при работе программы

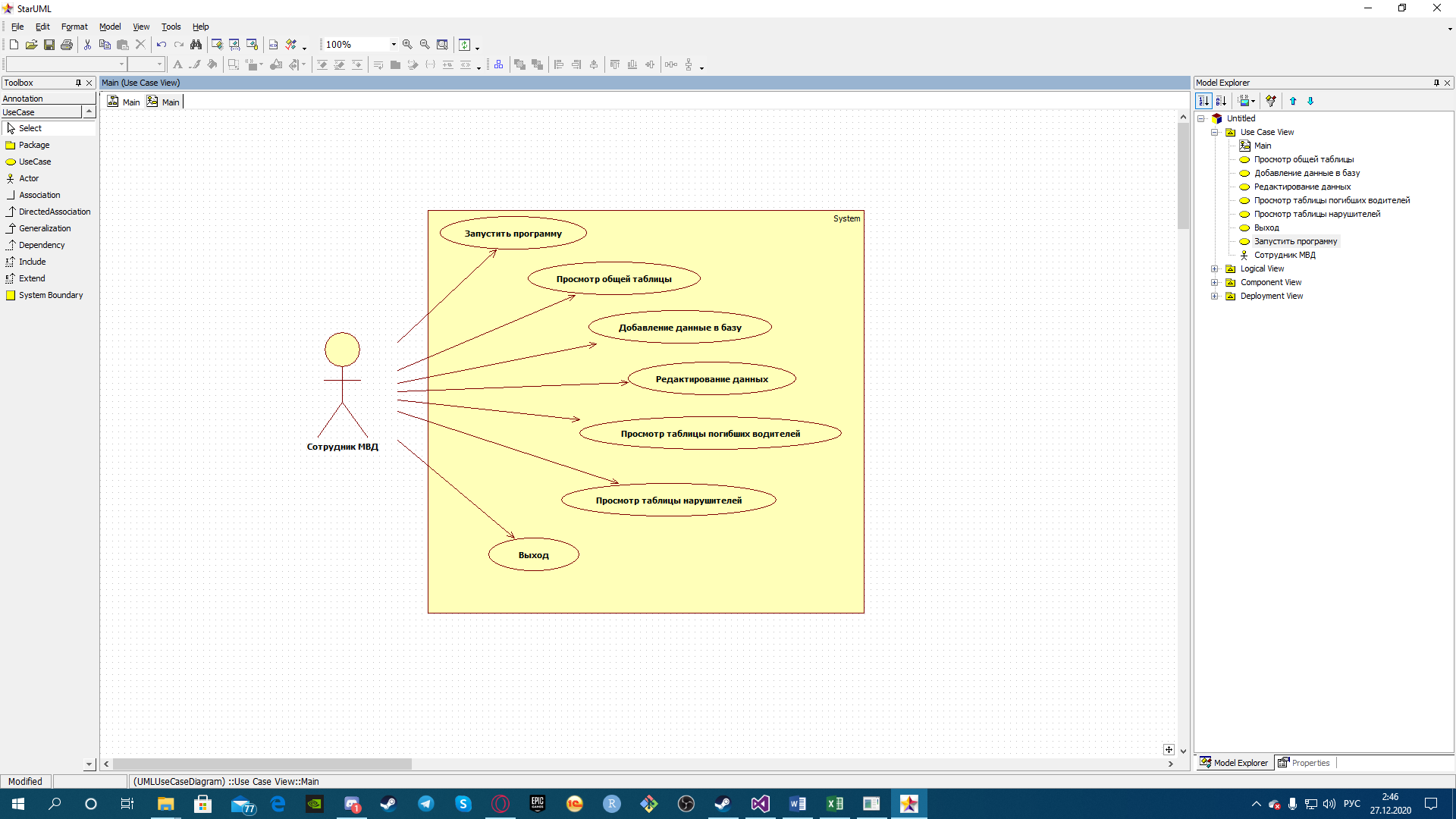


Рис.1 – Список вариантов использования, которые могут выполняться в проекте

# Каждый вариант использования должен быть отдельно описан текстом сценария

Вариант использования – Запустить программу

Пользователь инициализирует запуск программы и переходит на начальный экран (главное меню).

Вариант использования – Просмотр общей таблицы

Пользователь получает от системы таблицу данных, содержащую всю информацию о водителях (изначально таблица пуста).

Вариант использования – Добавление данных в таблицу.

Пользователь вносит в систему данные о водителе, а именно: ФИО, город, улицу, № дома, №квартиры, № телефона, число нарушений, № автомобиля, марку автомобиля, после чего система запоминает в себе данные и хранит их для последующего вызова.

Вариант использования – Редактирование данных:

Пользователь имеет возможность изменить данные того или иного водителя. Для этого ему необходимо ввести соответствующее ID. После это предоставляется возможность изменить все данные, указанные в вышеупомянутом варианте использования.

Вариант использования – Просмотр таблицы погибших водителей

Пользователь запрашивает у системы вывод информации о погибших водителях, на что система выдает ему таблицу, в которой имеются следующие данные: ID, ФИО и статус водителя.  
  
Вариант использования – Просмотр таблицы нарушителей.

Пользователь запрашивает у системы вывод информации о водителях, у которых имеется хотя бы одно нарушение, на что система выдает ему таблицу, в которой имеются следующие данные: ID, ФИО и количество нарушений.

Вариант использования – Выход.

Пользователь по нажатию на клавишу завершает работу программы и выходит из нее.

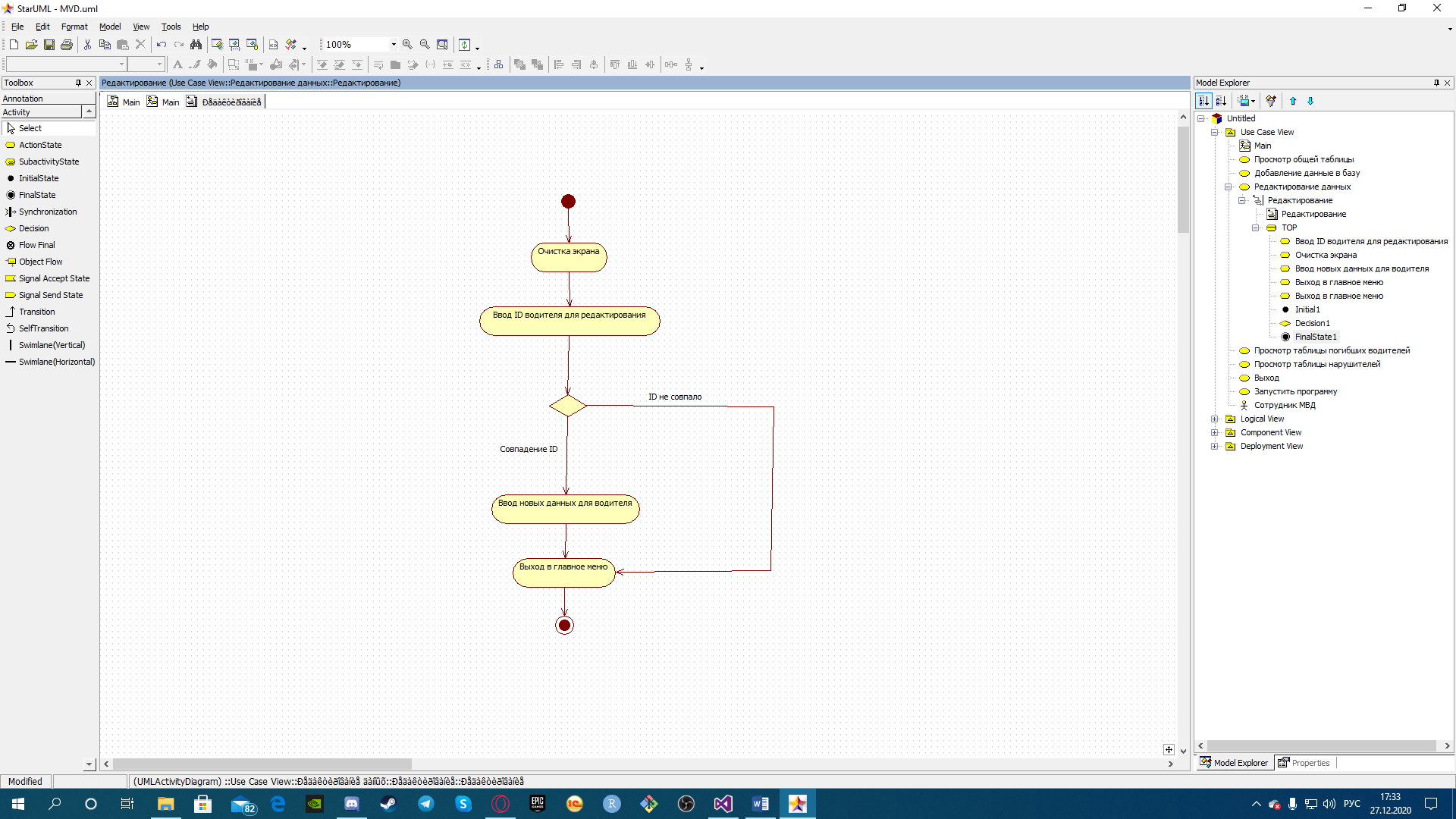
1. Не менее двух вариантов использования опишите диаграммами действий и приложите диаграммы к отчету  
     
     
   

Рис. 2 – Диаграмма действия: редактирование

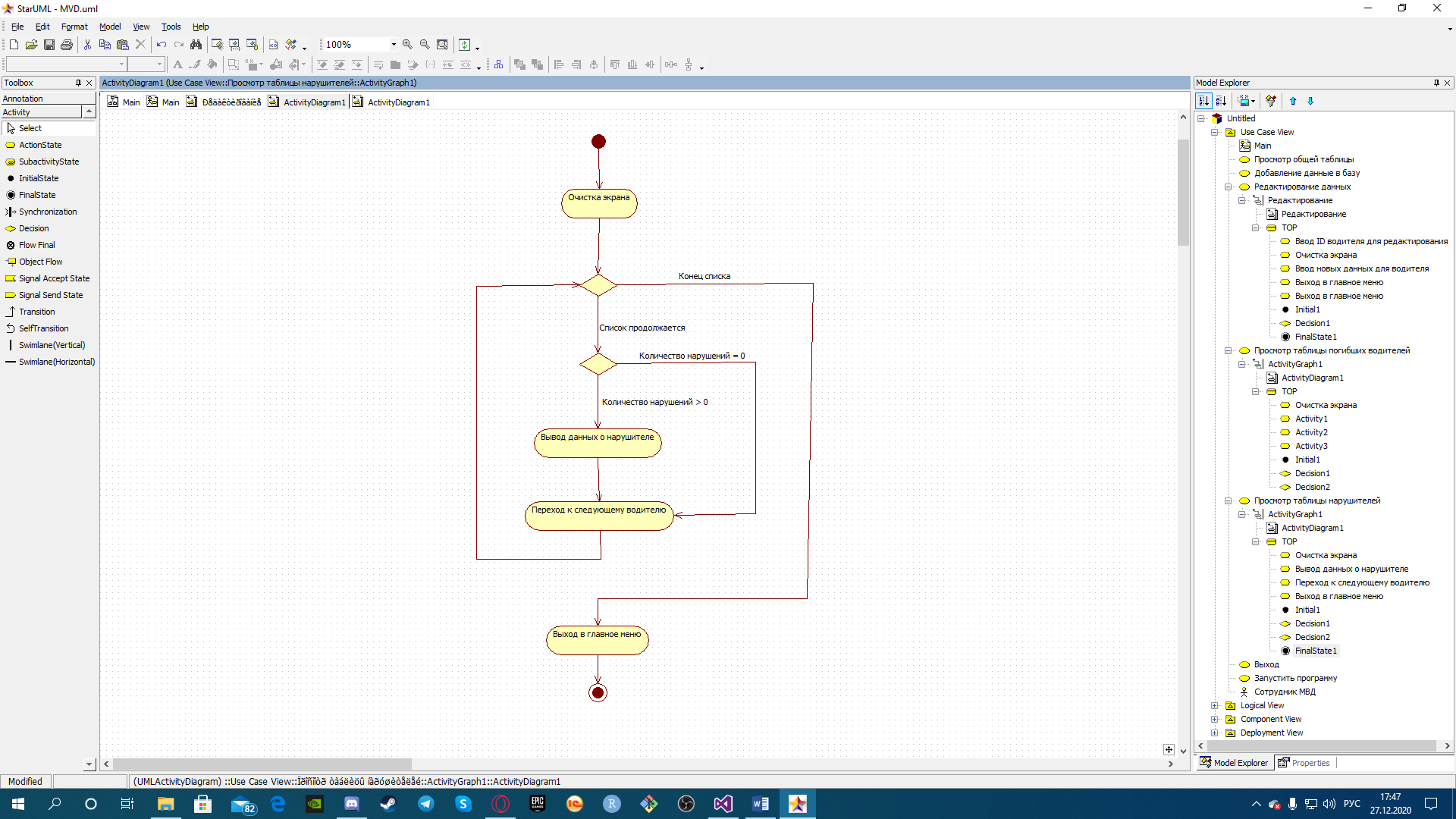


Рис.3 – Диаграмма действия: вывод таблицы нарушителей

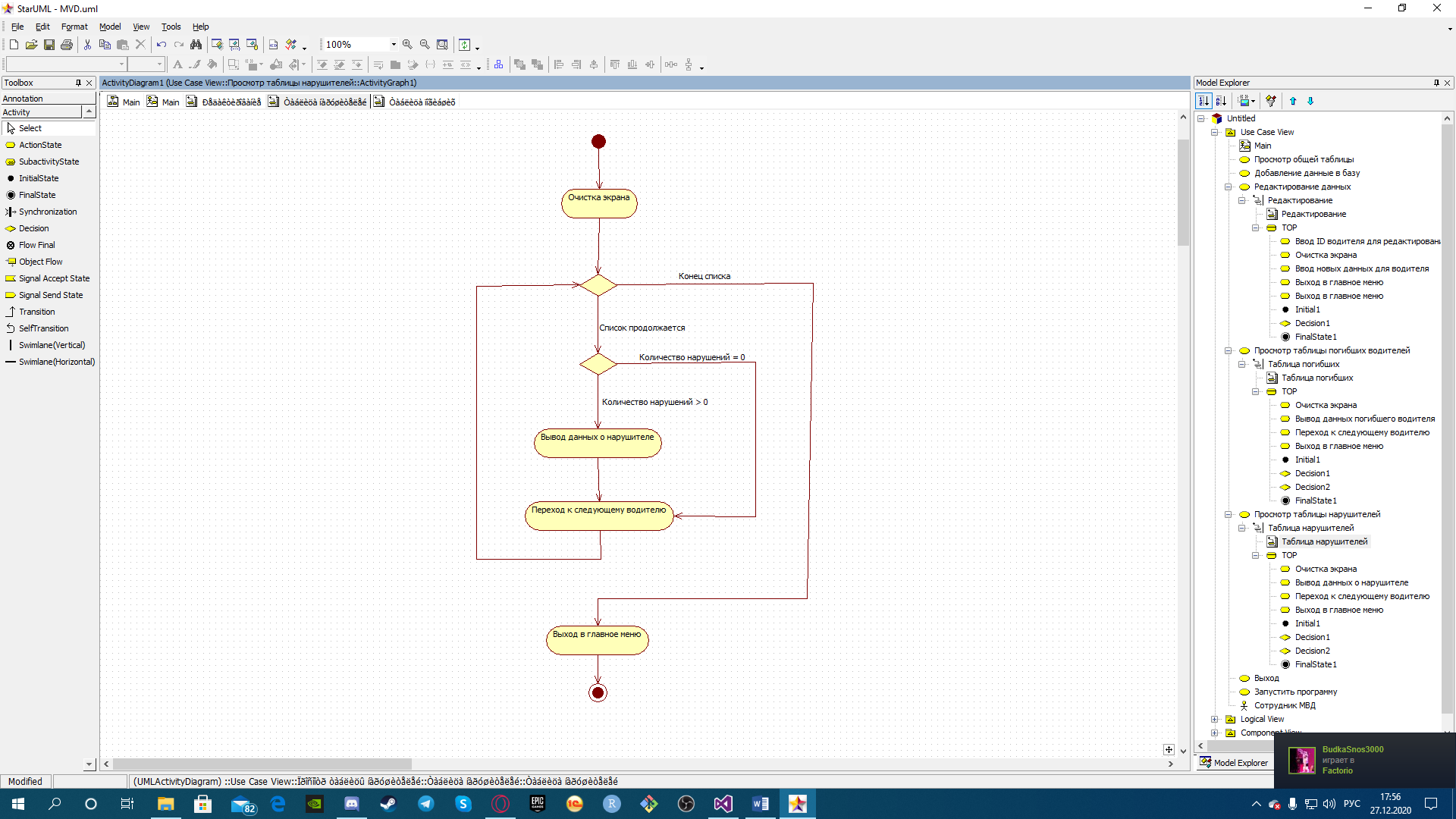


Рис. 4 – Диаграмма действия: вывод таблицы усопших водителей

1. Из описаний вариантов использования необходимо составить предварительный список существительных, которые, возможно, станут классами программы.   
     
   1) Главное меню  
   2) Экран добавления водителей  
   3) Фамилия  
   4) Имя  
   5) Отчество  
   6) Город  
   7) Улица  
   8) № дома  
   9) № квартиры  
   10) № телефона  
   11) Число нарушений  
   12) № автомобиля  
   13) Марка автомобиля  
   14) Жив/Мёртв водитель  
   15) Экран редактирования данных водителя  
   16) Вводимое ID для редактирования  
   17) Экран вывода погибших  
   18) Экран вывода нарушителей  
   19) Выход из программы
2. Список классов  
     
   1) Driver (Водитель)  
   2) DriverList (Список водителей)  
   3) DriverInputScreen (Экран добавления водителей)  
   4) InterfaceMenu (Основной интерфейс)

# По каждому классу составьте список атрибутов с пояснением предназначения каждого. Список атрибутов приложите к отчёту

1) Driver:  
ID; //Личный ID водителя  
Fam; //Фамилия  
Name; // Имя  
Otch; //Отчество  
City; // Город  
Street; //Улица  
NumberHome; //№ дома  
NumberFlat; //№ квартиры  
Phone; // Номер телефона  
Narushen; //Число нарушений  
NumberCar; //Номер автомобиля  
MarkCar; //Марка автомобиля   
Life; //Жив или мёртв водитель  
  
2) DriverList:

int sID; // ID для поиска совпадений в списке  
string EditFam; //Фамилия для редактирования  
string EditName; // Имя для редактирования  
string EditOtch; //Отчество для редактирования  
string EditCity; // Город для редактирования  
string EditStreet; //Улица для редактирования  
int EditNumberHome; //№ дома для редактирования  
int EditNumberFlat; //№ квартиры для редактирования  
long long EditPhone; // Номер телефона для редактирования  
int EditNarushen; //Число нарушений для редактирования  
int EditNumberCar; //Номер автомобиля для редактирования  
string EditMarkCar; //Марка автомобиля для редактирования  
string EditLife; //Жив или мёртв водитель для редактирования

int IDforEdit; // Вводимое ID для редактирования  
list <Driver\*> setPtrsDrivers; // указатели на класс водителей  
list <Driver\*>::iterator iter; //итератор  
  
3) DriverInputScreen:   
DriverList\* ptrDriverList; //Указатель на список водителей  
NewFam; // Новая фамилия  
NewName; // Новое имя  
NewOtch; // Новое отчество  
NewCity; // Новый город  
NewStreet; // Новая улица  
NewNumberHome; // Новый номер дома  
NewNumberFlat; // Новый номер квартиры  
NewPhone; // Новый номер телефона  
NewNarushen; // Новое число нарушений  
NewNumberCar; // Новый номер машины  
NewMarkCar; // Новая марка машины  
NewLife; // Состояние жизни (по умолчанию в список заносятся только живые водители)  
  
4) InterfaceMenu   
DriverList\* ptrDriverList; // Указатель на список водителей  
DriverInputScreen\* ptrDriverInputScreen; // Указатель на отображение «экрана», куда пользователь может ввести данные о новом водителе

1. Проанализируйте описания вариантов использования для выяснения того, какими сообщениями будут обмениваться классы.  
     
    1) Driver:   
    GetID(); // Возвращение значения личного ID водителя

GetFam(); // Возвращение значения фамилии  
GetName(); // Возвращение значения имени

GetOtch(); // Возвращение значения отчества

GetCity(); // Возвращение значения города

GetStreet(); // Возвращение значения улицы

GetNumberHome(); // Возвращение значения № дома

GetNumberFlat(); // Возвращение значения № квартиры

GetPhone(); // Возвращение значения № телефона

GetNarushen(); // Возвращение значения числа нарушений

GetNumberCar(); // Возвращение значения номера автомобиля

GetMarkCar(); // Возвращение значения марки автомобиля   
GetLife(); // Возвращение значения жив или мёртв водитель  
  
2) DriverList:   
~DriverList(); // деструктор (удаление водителей)

insertDriver(Driver\*); // добавить водителя в список

DisplayNarushen(); // вывод списка нарушителей

DisplayLife(); // вывод списка погибших

Editing(); // редактирование информации о водителе

display(); // вывод списка водителей

3) DriverInputScreen:  
DriverInputScreen(DriverList\* ptrDL) : ptrDriverList(ptrDL) // Производный указатель на список водителей

{

}

setDriver(); // добавить данные о водителе  
  
4) Interface  
InterfaceMenu(); // конструктор

~InterfaceMenu(); // деструктор

interact(); // основное меню

# Построить диаграмму классов

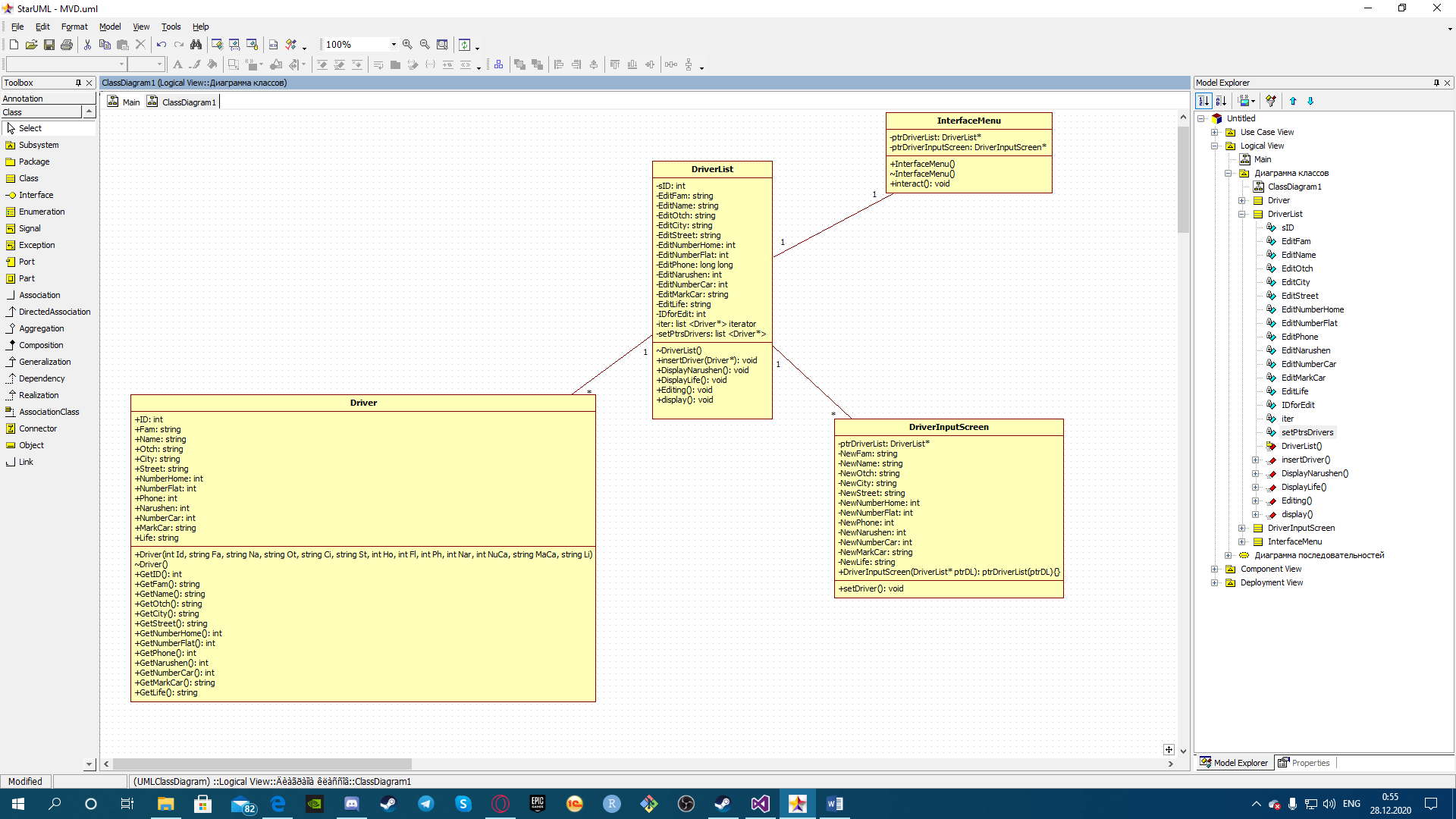


Рис. 5 – Диаграмма классов проекта

# Разработать диаграмму последовательностей UML, на которой отображается, каким образом события разворачиваются во времени.

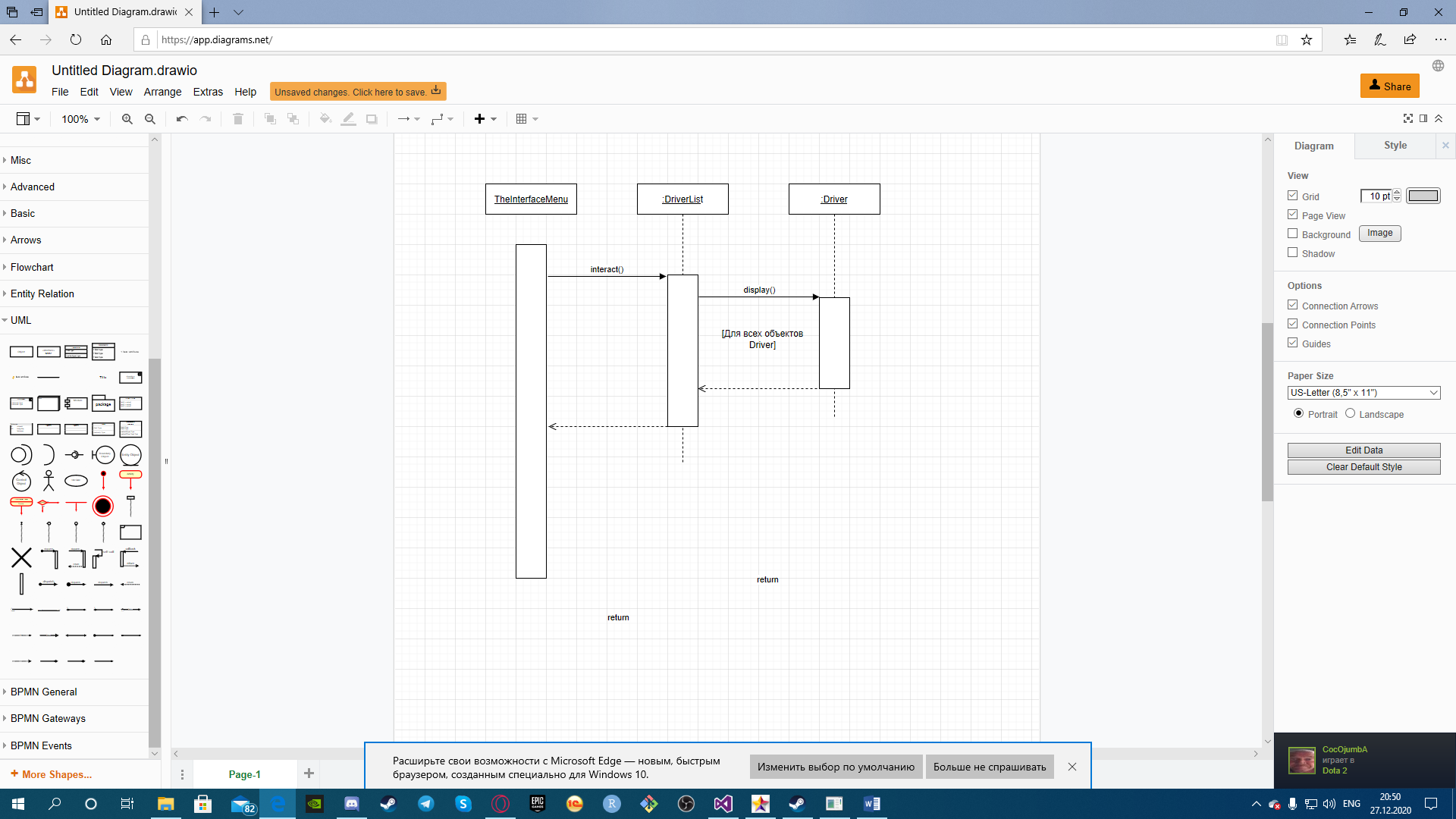


Рис. 6 – Вариант использования: запустить программу

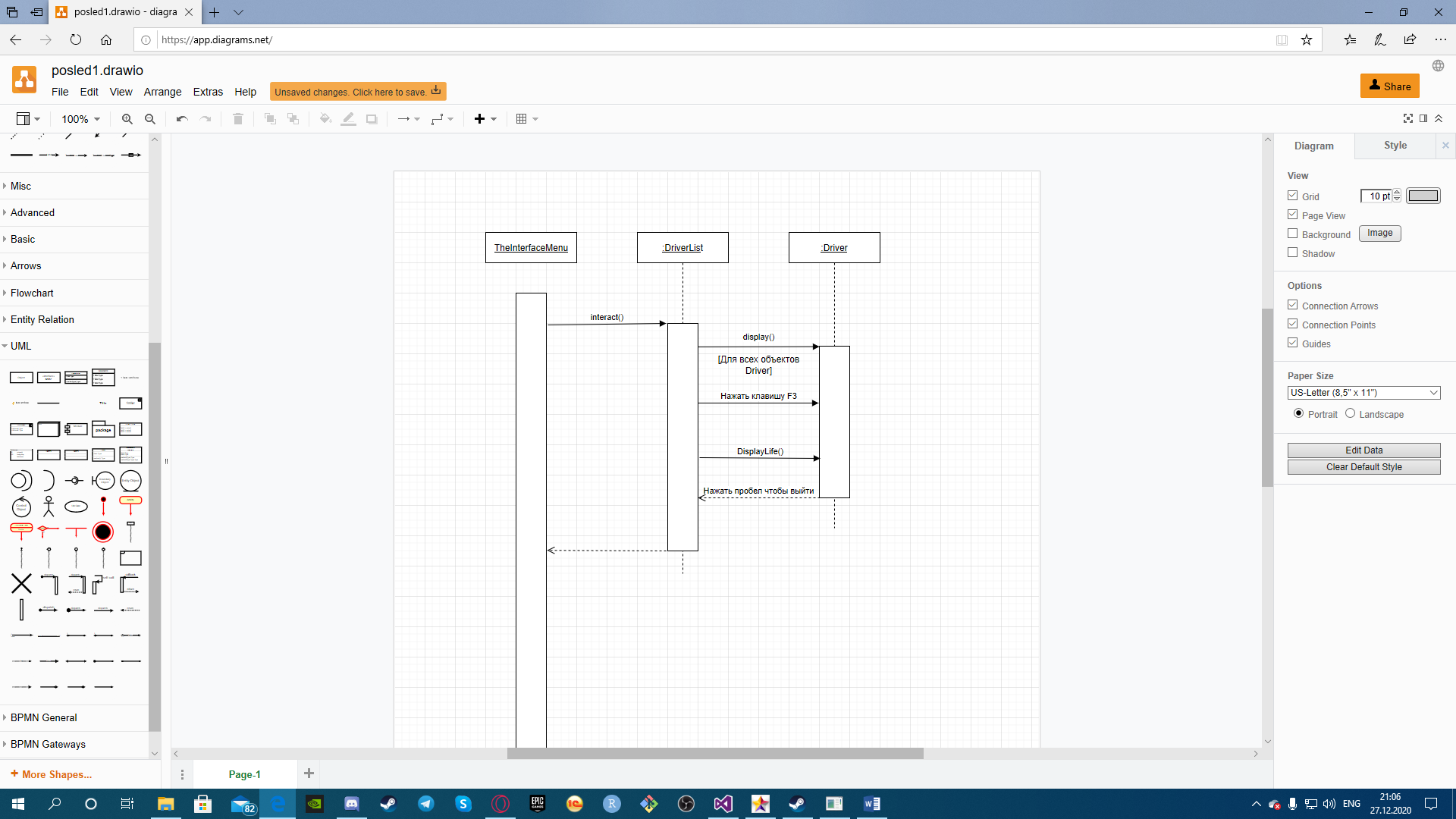


Рис. 7 - Вариант использования: просмотр таблицы погибших водителей

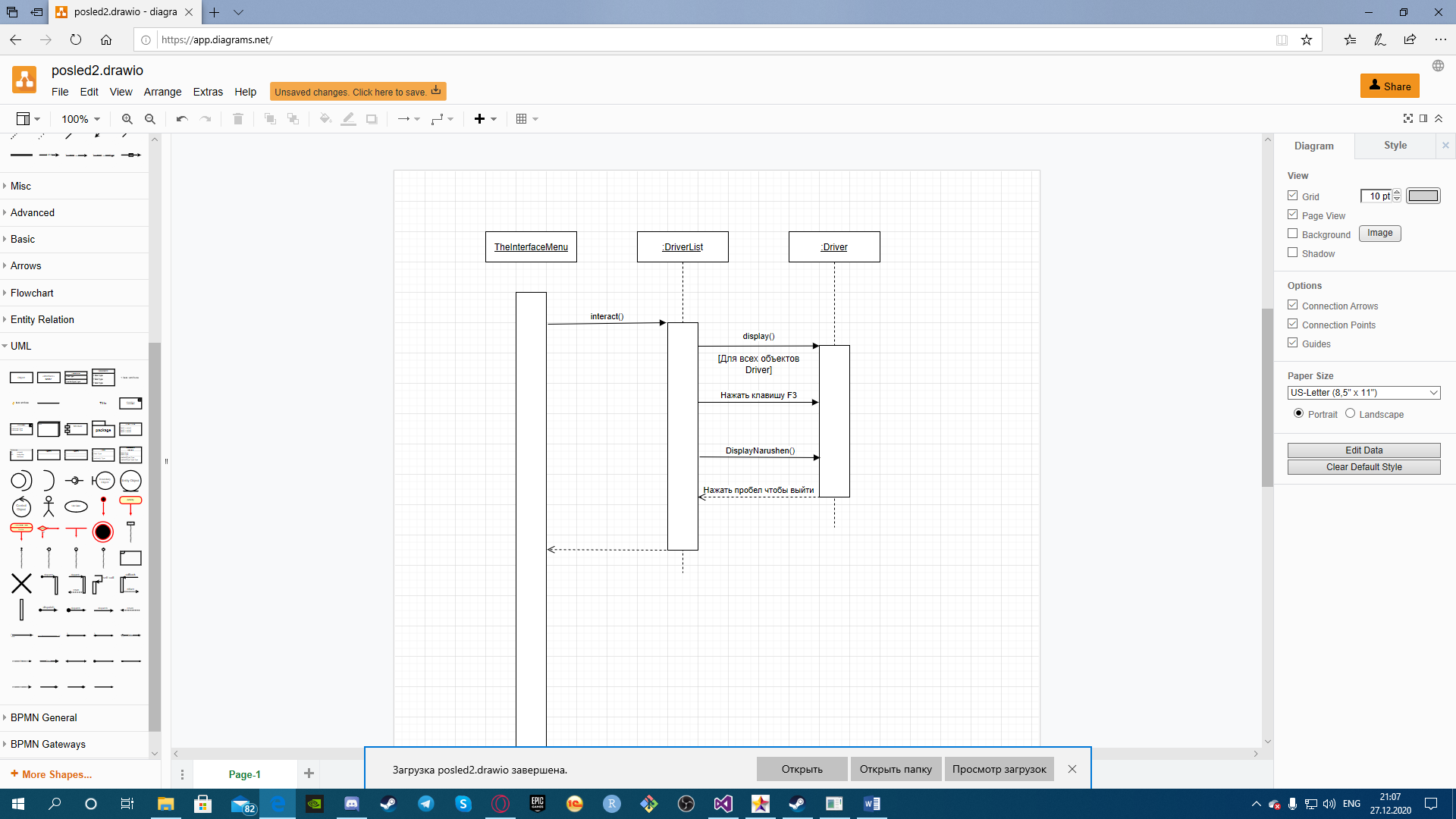


Рис. 8 – Вариант использования: просмотр таблицы нарушителей

# Написание кода программы. (.H) файлы

1. Заголовочный файл Driver.h

// Driver.h   
#ifndef \_\_CLASSDRIVER\_H\_\_

#define \_\_CLASSDRIVER\_H\_\_

#include <iostream>

#include <vector>

#include <list>

#include <string>

using namespace std;

//класс Driver

class Driver

{

public:

int ID; //Личный ID водителя

string Fam; //Фамилия

string Name; // Имя

string Otch; //Отчество

string City; // Город

string Street; //Улица

int NumberHome; //№ дома

int NumberFlat; //№ квартиры

long long Phone; // Номер телефона

int Narushen; //Число нарушений

int NumberCar; //Номер автомобиля

string MarkCar; //Марка автомобиля

string Life; //Жив или мёртв водитель

Driver(int Id, string Fa, string Na, string Ot, string Ci, string St, int Ho, int Fl, long long Ph, int Nar, int NuCa, string MaCa, string Li);

~Driver();

int GetID(); // Возвращение значения личного ID водителя

string GetFam(); // Возвращение значения фамилии

string GetName(); // Возвращение значения имени

string GetOtch(); // Возвращение значения отчества

string GetCity(); // Возвращение значения города

string GetStreet(); // Возвращение значения улицы

int GetNumberHome(); // Возвращение значения № дома

int GetNumberFlat(); // Возвращение значения № квартиры

long long GetPhone(); // Возвращение значения № телефона

int GetNarushen(); // Возвращение значения числа нарушений

int GetNumberCar(); // Возвращение значения номера автомобиля

string GetMarkCar(); // Возвращение значения марки автомобиля

string GetLife(); // Возвращение значения жив или мёртв водитель

};

#endif

1. Заголовочный файл DriverInputScreen.h

// DriverInputScreen.h

#ifndef \_\_CLASSDRIVERINPUTSCREEN\_H\_\_

#define \_\_CLASSDRIVERINPUTSCREEN\_H\_\_

#include <iostream>

#include <vector>

#include <list>

#include <string>

#include "ClassDriverList.h"

using namespace std;

//класс DriverInputScreen

//класс DriverInputScreen. Это класс, отвечающий за отображение «экрана»,

//куда пользователь может ввести данные о новом водителе:

class DriverInputScreen

{

private:

DriverList\* ptrDriverList; //Указатель на список водителей

string NewFam; // Новая фамилия

string NewName; // Новое имя

string NewOtch; // Новое отчество

string NewCity; // Новый город

string NewStreet; // Новая улица

int NewNumberHome; // Новый номер дома

int NewNumberFlat; // Новый номер квартиры

long long NewPhone; // Новый номер телефона

int NewNarushen; // Новое число нарушений

int NewNumberCar; // Новый номер машины

string NewMarkCar; // Новая марка машины

string NewLife; // Состояние жизни (по умолчанию в список заносятся только живые водители)

public:

DriverInputScreen(DriverList\* ptrDL) : ptrDriverList(ptrDL) // Производный указатель на список водителей

{

}

void setDriver(); // добавить данные о водителе

};

#endif

1. Заголовочный файл DriverList.h

// DriverList.h  
#ifndef \_\_CLASSDRIVERLIST\_H\_\_

#define \_\_CLASSDRIVERLIST\_H\_\_

#include <iostream>

#include <vector>

#include <list>

#include <string>

#include "ClassDriver.h"

using namespace std;

//класс DriverList

//класс DriverList — список всех водителей.

//Он содержит множество указателей на класс Driver

// и оперирует ими при выводе

class DriverList

{

private:

int sID; // ID для поиска совпадений в списке

string EditFam; //Фамилия для редактирования

string EditName; // Имя для редактирования

string EditOtch; //Отчество для редактирования

string EditCity; // Город для редактирования

string EditStreet; //Улица для редактирования

int EditNumberHome; //№ дома для редактирования

int EditNumberFlat; //№ квартиры для редактирования

long long EditPhone; // Номер телефона для редактирования

int EditNarushen; //Число нарушений для редактирования

int EditNumberCar; //Номер автомобиля для редактирования

string EditMarkCar; //Марка автомобиля для редактирования

string EditLife; //Жив или мёртв водитель для редактирования

int IDforEdit; // Вводимое ID для редактирования

list <Driver\*> setPtrsDrivers; // указатели на класс водителей

list <Driver\*>::iterator iter; //итератор

public:

~DriverList(); // деструктор (удаление водителей)

void insertDriver(Driver\*); // добавить водителя в список

void DisplayNarushen(); // вывод списка нарушителей

void DisplayLife(); // вывод списка погибших

void Editing(); // редактирование информации о водителе

void display(); // вывод списка водителей

};

#endif

1. Заголовочный файл GlobalMethods.h

#include <iostream>

#include <vector>

#include <list>

#include <string>

using namespace std;

// глобальные методы

void getaLine(string& inStr); // получение строки текста

char getaChar(); // получение символа

void enterProbel(); // нажатие пробела

void color(int); //цвет текста

int CheckInt(int i); // проверка на неотрицательность переменной типа unsigned int

long long CheckLong(long long l); // проверка на неотрицательность переменной типа unsigned long long

1. Заголовочный файл InterfaceMenu.h

// InterfaceMenu.h  
#ifndef \_\_CLASSINTERFACEMENU\_H\_\_

#define \_\_CLASSINTERFACEMENU\_H\_\_

#include <iostream>

#include <vector>

#include <list>

#include <string>

#include "ClassDriverList.h"

#include "ClassDriverInputScreen.h"

using namespace std;

//класс InterfaceMenu

class InterfaceMenu

{

private:

DriverList\* ptrDriverList; // Указатель на список водителей

DriverInputScreen\* ptrDriverInputScreen; // Указатель на отображение «экрана», куда пользователь может ввести данные о новом водителе

public:

InterfaceMenu(); // конструктор

~InterfaceMenu(); // деструктор

void interact(); // основное меню

};

#endif

1. Написание исходных файлов. (.cpp) файлы
2. Исходный файл GlobalMethods.cpp

// GlobalMethods.cpp  
#include "stdafx.h"

#include <iostream>

#include <windows.h>

#include <conio.h>

#include <iomanip>

#include "GlobalMethods.h"

using namespace std;

void getaLine(string& inStr) // получение строки текста

{

char temp[21];

cin.get(temp, 20, '\n');

cin.ignore(20, '\n'); //число пропускаемых символов и символ разделения

inStr = temp;

}

char getaChar() // получение символа

{

char ch = cin.get();

cin.ignore(20, '\n'); //число пропускаемых символов и символ разделения

return ch;

}

void enterProbel() //

{

int key = 1;

cout << "Нажмите пробел для продолжения";

while (key)

{

char ch = \_getch();

int code = static\_cast<int>(ch);

if (ch == 32)

key = 0;

}

}

void color(int color) // Кейсы с цветом текста

{

HANDLE hwnd = GetStdHandle(STD\_OUTPUT\_HANDLE);

switch (color)

{

case 0://Цвет текста белый

SetConsoleTextAttribute(hwnd, FOREGROUND\_INTENSITY | FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);

break;

case 1://Цвет текста Зеленый

SetConsoleTextAttribute(hwnd, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

break;

case 2://Цвет текста Красный

SetConsoleTextAttribute(hwnd, FOREGROUND\_RED | FOREGROUND\_INTENSITY);

break;

}

}

int CheckInt(int n) // проверка числа на неотрицательность

{

if (n < 0)

{

cout << "Введите неотрицательное число\n";

}

else return n;

}

long long CheckLong(long long l) // проверка числа на неотрицательность

{

if (l < 0)

{

cout << "Введите неотрицательное число\n";

}

else return l;

}

1. Исходный файл Driver.cpp

// Driver.cpp  
#include "stdafx.h"

#include <iostream>

#include <windows.h>

#include <conio.h>

#include <iomanip>

#include "ClassDriver.h"

using namespace std;

//методы класса Driver

//Конструктор

Driver::Driver(int Id, string Fa, string Na, string Ot, string Ci, string St, int Ho, int Fl, long long Ph, int Nar, int NuCa, string MaCa, string Li) :

ID(Id), Fam(Fa), Name(Na), Otch(Ot), City(Ci), Street(St), NumberHome(Ho), NumberFlat(Fl), Phone(Ph), Narushen(Nar), NumberCar(NuCa), MarkCar(MaCa), Life(Li)

{

}

Driver::~Driver() // деструктор

{

}

int Driver::GetID() //геттер возвращает id

{

return ID;

}

string Driver::GetFam() //геттер возвращает фамилию

{

return Fam;

}

string Driver::GetName() //геттер возвращает имя

{

return Name;

}

string Driver::GetOtch() //геттер возвращает отчество

{

return Otch;

}

string Driver::GetCity() //геттер возвращает город

{

return City;

}

string Driver::GetStreet() //геттер возвращает улицу

{

return Street;

}

int Driver::GetNumberHome() //геттер возвращает номер дома

{

return NumberHome;

}

int Driver::GetNumberFlat() //геттер возвращает номер квартиры

{

return NumberFlat;

}

long long Driver::GetPhone() //геттер возвращает номер телефона

{

return Phone;

}

int Driver::GetNarushen() //геттер возвращает количество нарушений

{

return Narushen;

}

int Driver::GetNumberCar() //геттер возвращает номер машины

{

return NumberCar;

}

string Driver::GetMarkCar() //геттер возвращает марку машины

{

return MarkCar;

}

string Driver::GetLife() //геттер возвращает значение о состоянии водителя(жив/мёртв)

{

return Life;

}

1. Исходный файл DriverInputScreen.cpp

// DriverInputScreen.cpp  
#include "stdafx.h"

#include <iostream>

#include <windows.h>

#include <conio.h>

#include <iomanip>

#include "GlobalMethods.h"

#include "ClassDriverInputScreen.h"

using namespace std;

int NewID; //Личный ID каждого водителя. При добавлении нового водителя увеличивается на 1

//метод класса DriverInputScreen

void DriverInputScreen::setDriver() // добавить данные о водителе

{

color(0);

cout << "Фамилия: ";

getaLine(NewFam);

cout << "Имя: ";

getaLine(NewName);

cout << "Отчество: ";

getaLine(NewOtch);

cout << "Город: ";

getaLine(NewCity);

cout << "Улица: ";

getaLine(NewStreet);

cout << "Номер дома: ";

do

{

cin >> NewNumberHome;

CheckInt(NewNumberHome);

cin.ignore(25, '\n');

} while (NewNumberHome < 0);

cout << "Номер квартиры: ";

do

{

cin >> NewNumberFlat;

CheckInt(NewNumberFlat);

cin.ignore(25, '\n');

} while (NewNumberFlat < 0);

cout << "Телефон: ";

do

{

cin >> NewPhone;

CheckLong(NewPhone);

cin.ignore(25, '\n');

} while (NewPhone < 0);

cout << "Число нарушений: ";

do

{

cin >> NewNarushen;

CheckInt(NewNarushen);

cin.ignore(25, '\n');

} while (NewNarushen < 0);

cout << "Номер автомобиля:";

do

{

cin >> NewNumberCar;

CheckInt(NewNumberCar);

cin.ignore(25, '\n');

} while (NewNumberCar < 0);

cout << "Марка автомобиля: ";

getaLine(NewMarkCar);

NewID = NewID++;

NewLife = ("Жив");

Driver\* ptrDriver = new Driver(NewID, NewFam, NewName, NewOtch, NewCity, NewStreet, NewNumberHome, NewNumberFlat, NewPhone, NewNarushen, NewNumberCar, NewMarkCar, NewLife); // создать жильца

ptrDriverList->insertDriver(ptrDriver); // занести в список водителей

}

//---------------------------------------------------------

1. Исходный файл DriverList.cpp

// DriverList.cpp

#include "stdafx.h"

#include <iostream>

#include <windows.h>

#include <conio.h>

#include <iomanip>

#include "ClassDriverList.h"

#include "GlobalMethods.h"

using namespace std;

//методы класса DriverList

DriverList::~DriverList() // деструктор

{

while (!setPtrsDrivers.empty()) // удаление всех водителей,

{ // удаление указателей из контейнера

iter = setPtrsDrivers.begin();

delete \*iter;

setPtrsDrivers.erase(iter);

}

}

//---------------------------------------------------------

void DriverList::insertDriver(Driver\* ptrD)

{

setPtrsDrivers.push\_back(ptrD); // вставка нового водителя в список

}

//--------------------------------------------------------

void DriverList::Editing()

{

color(0);

cout << "Введите ID водителя, данные которого вы хотите изменить: " << endl;

cin >> IDforEdit;

iter = setPtrsDrivers.begin(); // от начала списка

while (iter != setPtrsDrivers.end()) // до конца списка

{ // поиск водителя в списке (сравниваем введенное ID с ID в списке)

sID = (\*iter)->GetID();

if (IDforEdit == sID) // сравниваем введенное id со всеми id в списке

{

cout << "Введите новую фамилию: ";

cin >> EditFam;

(\*iter)->Fam = EditFam;

cout << "Введите новое имя: ";

cin >> EditName;

(\*iter)->Name = EditName;

cout << "Введите новое отчество: ";

cin >> EditOtch;

(\*iter)->Otch = EditOtch;

cout << "Введите новый город: ";

cin >> EditCity;

(\*iter)->City = EditCity;

cout << "Введите новую улицу: ";

cin >> EditStreet;

(\*iter)->Street = EditStreet;

cout << "Введите новый номер дома: ";

do

{

cin >> EditNumberHome;

CheckInt(EditNumberHome);

} while (EditNumberHome < 0);

(\*iter)->NumberHome = EditNumberHome;

cout << "Введите новый номер квартиры: ";

do

{

cin >> EditNumberFlat;

CheckInt(EditNumberFlat);

} while (EditNumberFlat < 0);

(\*iter)->NumberFlat = EditNumberFlat;

cout << "Введите новый номер телефона: ";

do

{

cin >> EditPhone;

CheckLong(EditPhone);

} while (EditPhone < 0);

(\*iter)->Phone = EditPhone;

cout << "Введите новое число нарушений: ";

do

{

cin >> EditNarushen;

CheckLong(EditNarushen);

} while (EditNarushen < 0);

(\*iter)->Narushen = EditNarushen;

cout << "Введите новый номер автомобиля:";

do

{

cin >> EditNumberCar;

CheckLong(EditNumberCar);

} while (EditNumberCar < 0);

(\*iter)->NumberCar = EditNumberCar;

cout << "Введите новую марку автомобиля: ";

cin >> EditMarkCar;

(\*iter)->MarkCar = EditMarkCar;

cout << "Жив ли водитель? (Y(да) или N(нет))" << endl;

do // Проверка на введенное значение

{

cin >> EditLife;

if (EditLife == "Y")

{

(\*iter)->Life = "Жив";

}

else if (EditLife == "N")

{

(\*iter)->Life = "Мёртв";

}

else

{

cout << "Для ввода доступны только буквы Y и N";

}

} while ((EditLife != "Y") && (EditLife != "N"));

}

iter++;

}

void enterProbel(); // По нажатию пробела возвращаемся к главному меню

}

void DriverList::DisplayLife() // Экран с погибшими водителями

{

string tLife; // Перенная необходимая для проверки на жизненное состояние (жив водитель или нет)

color(1);

cout << setw(100) << "--------------------------" << endl;

cout << setw(100) << "|СПИСОК УМЕРШИХ ВОДИТЕЛЕЙ|" << endl;

cout << setw(100) << "--------------------------" << endl;

color(2);

cout << "\n\n-----------------------------------------------------------------------" << endl;

cout << "|| ID || Фамилия || Имя || Отчество || Жив/Мёртв ||" << endl;

cout << "-----------------------------------------------------------------------" << endl;

if (setPtrsDrivers.empty()) // если список водителей пуст

cout << "\n\n\*\*\*На данный момент список пуст. Добавьте данные.\*\*\*\n" << endl; // выводим запись, что он пуст)

else

{

iter = setPtrsDrivers.begin();

while (iter != setPtrsDrivers.end()) // распечатываем всех водителей

{

tLife = (\*iter)->GetLife(); // получаем значение состояния жизни каждого водителя

if (tLife != "Жив") // Если водитель не жив...

{

color(0);

cout << "||"

<< setw(4) << (\*iter)->GetID() << "||"

<< setw(15) << (\*iter)->GetFam() << "||"

<< setw(13) << (\*iter)->GetName() << "||"

<< setw(16) << (\*iter)->GetOtch() << "||"

<< setw(11) << (\*iter)->GetLife() << "||" << endl;

\*iter++;

color(2);

cout << "-----------------------------------------------------------------------" << endl;

}

else \*iter++;

}

}

color(0);

cout << "\n\n";

void enterProbel(); // По нажатию пробела возвращаемся к главному меню

}

//--------------------------------------------------------

void DriverList::DisplayNarushen() // Экран с нарушителями

{

int tNarushen; // Перенная необходимая для проверки количество нарушений

color(1);

cout << setw(100) << "-------------------------------" << endl;

cout << setw(100) << "|РЕЗУЛЬТАТЫ ПОИСКА НАРУШИТЕЛЕЙ|" << endl;

cout << setw(100) << "-------------------------------" << endl;

color(2);

cout << "\n\n-----------------------------------------------------------------------------------" << endl;

cout << "|| ID || Фамилия || Имя || Отчество || Число нарушений ||" << endl;

cout << "-----------------------------------------------------------------------------------" << endl;

if (setPtrsDrivers.empty()) // если список водителей пуст

cout << "\n\n\*\*\*На данный момент список пуст. Добавьте данные.\*\*\*\n" << endl; // выводим запись, что он пуст)

else

{

iter = setPtrsDrivers.begin();

while (iter != setPtrsDrivers.end()) // распечатываем всех водителей

{

tNarushen = (\*iter)->GetNarushen(); // получаем значение количества нарушений каждого водителя

if (tNarushen > 0) // если есть хоть одно нарушение...

{

color(0);

cout << "||"

<< setw(4) << (\*iter)->GetID() << "||"

<< setw(15) << (\*iter)->GetFam() << "||"

<< setw(13) << (\*iter)->GetName() << "||"

<< setw(16) << (\*iter)->GetOtch() << "||"

<< setw(23) << (\*iter)->GetNarushen() << "||" << endl;

\*iter++;

color(2);

cout << "-----------------------------------------------------------------------------------" << endl;

}

else \*iter++;

}

}

color(0);

cout << "\n\n";

void enterProbel(); // По нажатию пробела возвращаемся к главному меню

}

//--------------------------------------------------------

void DriverList::display() // вывод списка водителей (основная таблица)

{

color(2);

cout << "\n\n";

cout << "-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------" << endl;

cout << "|| ID || Фамилия || Имя || Отчество || Город || Улица || № дома || № квартиры || Телефон || Число нарушений || № авто || Марка || Жив/Мёртв ||" << endl;

cout << "-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------" << endl;

color(0);

if (setPtrsDrivers.empty()) // если список водителей пуст

cout << "\n\n\*\*\*На данный момент список пуст. Добавьте данные.\*\*\*\n" << endl; // выводим запись, что он пуст

else

{

iter = setPtrsDrivers.begin();

while (iter != setPtrsDrivers.end()) // распечатываем всех водителей

{

color(0);

cout << "||"

<< setw(5) << (\*iter)->GetID() << "||"

<< setw(13) << (\*iter)->GetFam() << "||"

<< setw(13) << (\*iter)->GetName() << "||"

<< setw(16) << (\*iter)->GetOtch() << "||"

<< setw(15) << (\*iter)->GetCity() << "||"

<< setw(19) << (\*iter)->GetStreet() << "||"

<< setw(8) << (\*iter)->GetNumberHome() << "||"

<< setw(12) << (\*iter)->GetNumberFlat() << "||"

<< setw(13) << (\*iter)->GetPhone() << "||"

<< setw(17) << (\*iter)->GetNarushen() << "||"

<< setw(10) << (\*iter)->GetNumberCar() << "||"

<< setw(17) << (\*iter)->GetMarkCar() << "||"

<< setw(11) << (\*iter)->GetLife() << "||" << endl;

\*iter++;

color(2);

cout << "-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------" << endl;

}

}

}

//---------------------------------------------------------

1. Исходный файл InterfaceMenu.cpp

// InterfaceMenu.cpp  
#include "stdafx.h"

#include <iostream>

#include <windows.h>

#include <conio.h>

#include <iomanip>

#include "ClassInterfaceMenu.h"

#include "GlobalMethods.h"

using namespace std;

////////////////методы класса InterfaceMenu

InterfaceMenu::InterfaceMenu() // конструктор

{

ptrDriverList = new DriverList;

}

//---------------------------------------------------------

InterfaceMenu::~InterfaceMenu() // деструктор

{

delete ptrDriverList;

}

//---------------------------------------------------------

void InterfaceMenu::interact() // наше главное меню из которого мы переходим к другим параметрам (таблицам)

{

while (true)

{ // горячие клавиши

system("cls");

color(1);

cout << "--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------" << endl;

cout << "|| ";

color(2);

cout << "F1.Добавить данные ";

color(1);

cout << " || ";

color(2);

cout << "F2.Редактирование информации ";

color(1);

cout << " || ";

color(2);

cout << "F3.Список погибших";

color(1);

cout << " || ";

color(2);

cout << "F4.Поиск нарушителей ";

color(1);

cout << " || ";

color(2);

cout << "F5.Выход ";

color(1);

cout << " ||" << endl;

cout << "--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------" << endl;

color(0);

ptrDriverList->display(); // вывод основной таблицы на начальном экране

switch (\_getch())

{

case 59://F1.Добавить данные

system("cls"); // очистка экрана

ptrDriverInputScreen = new DriverInputScreen(ptrDriverList);

ptrDriverInputScreen->setDriver();

delete ptrDriverInputScreen;

break;

case 60://F2 Редактировать данные

system("cls");

ptrDriverList->Editing();

break;

case 61://F3 Список умерших водителей

system("cls");

ptrDriverList->DisplayLife();

system("pause");

break;

case 62://F4 Поиск нарушителей

system("cls");

ptrDriverList->DisplayNarushen();

system("pause");

break;

case 63://F5 Выход

return;

break;

}

}

}

1. Исходный файл OurProject.cpp

// OurProject.cpp

#include "stdafx.h"

#include "ClassInterfaceMenu.h"

int main()

{

setlocale(LC\_ALL, "Russian"); // задаём русский текст

system("chcp 1251"); // настраиваем кодировку консоли

InterfaceMenu TheInterfaceMenu;

TheInterfaceMenu.interact(); // Вывод меню

return 0;

}

# Руководство пользователя

Наша программа предназначена для заполнения информации о водителях. Система запоминает введенные Вами данные, а после выдает их, если ей следует на это команда.

При запуске программы Вы попадаете в главное меню.



Рис.9 – Вид на главное меню программы

В данном меню в верхней части экрана находятся клавиши быстрого доступа (F1, F2, F3, F4, F5), нажимая которые Вы будете совершать соответствующее клавише действие.

Для заполнения таблицы необходимо нажать клавишу F1(Добавить данные).

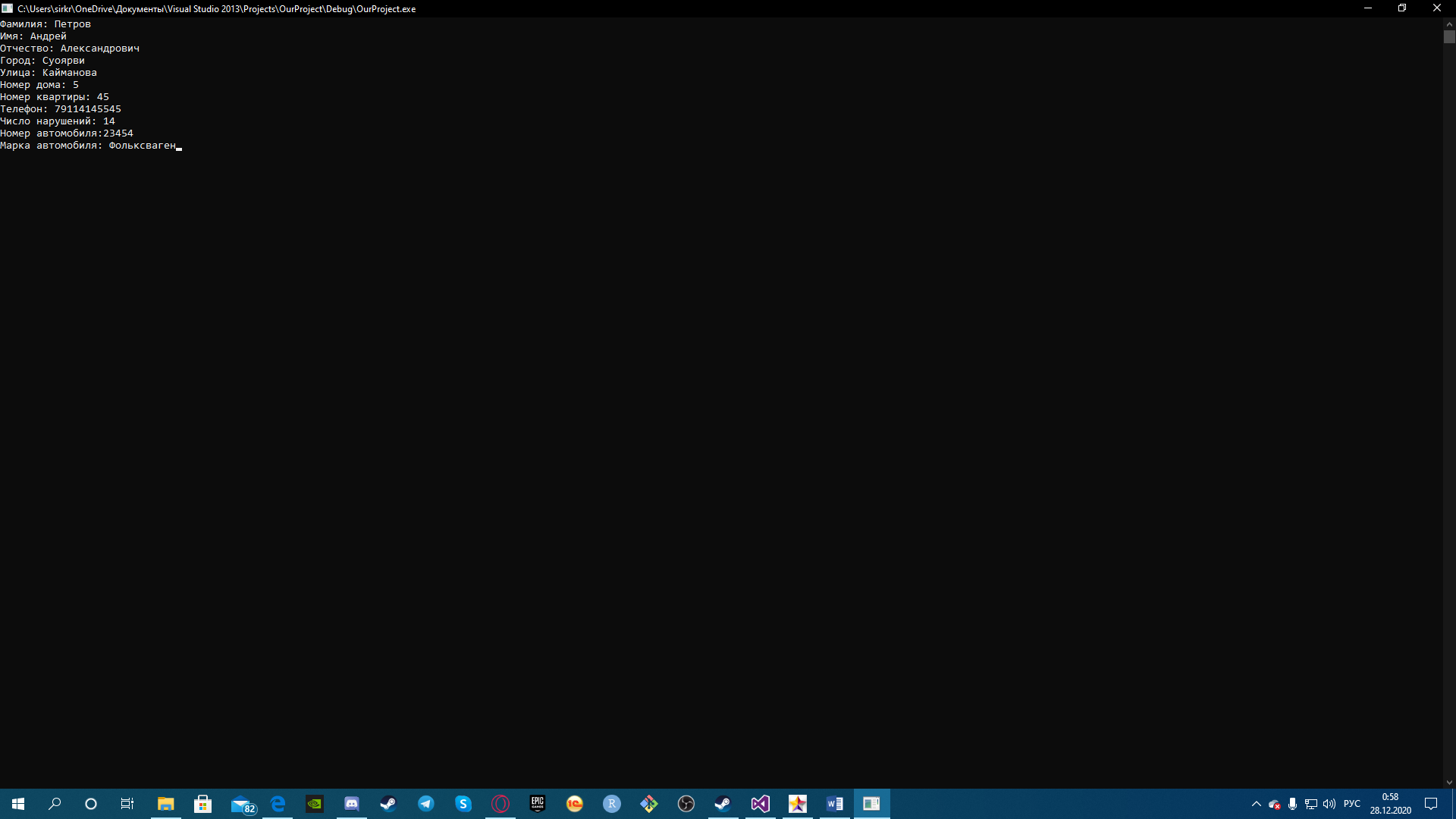


Рис. 10 – Режим заполнения таблицы (при нажатии F1)

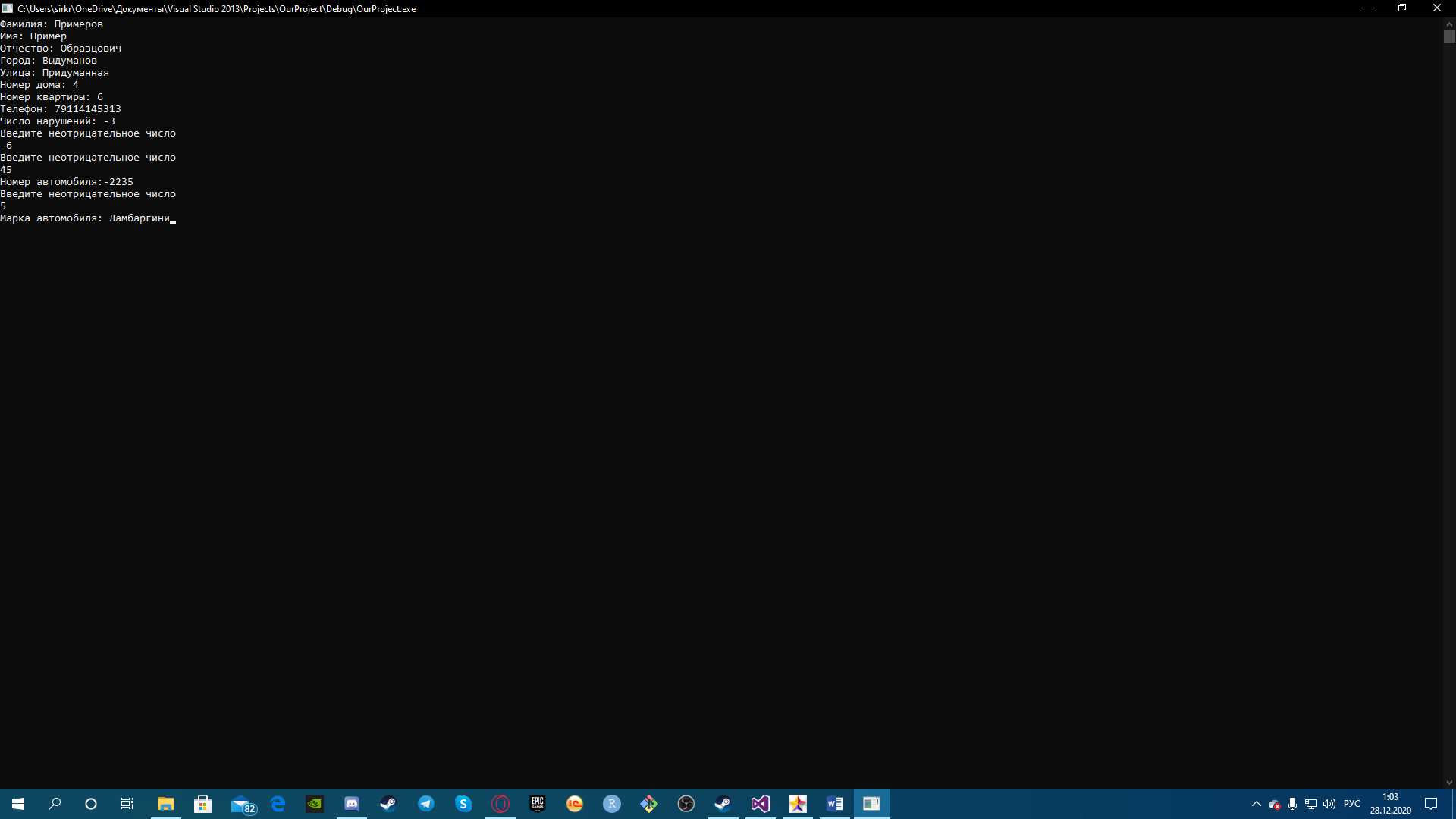


Рис. 11 – Пример проверки на отрицательные числа в графах числа нарушений и номера автомобиля

Добавляя новых водителей в системы, Вы будете получать всю актуальную информацию о них в виде таблицы на главном меню.



Рис. 12 – Пример вида главного меню при заполненной таблице

При необходимости редактирования водителя Вы можете нажать на горячую клавишу F2. После этого Вам необходимо ввести ID человека, которое Вам необходимо изменить.

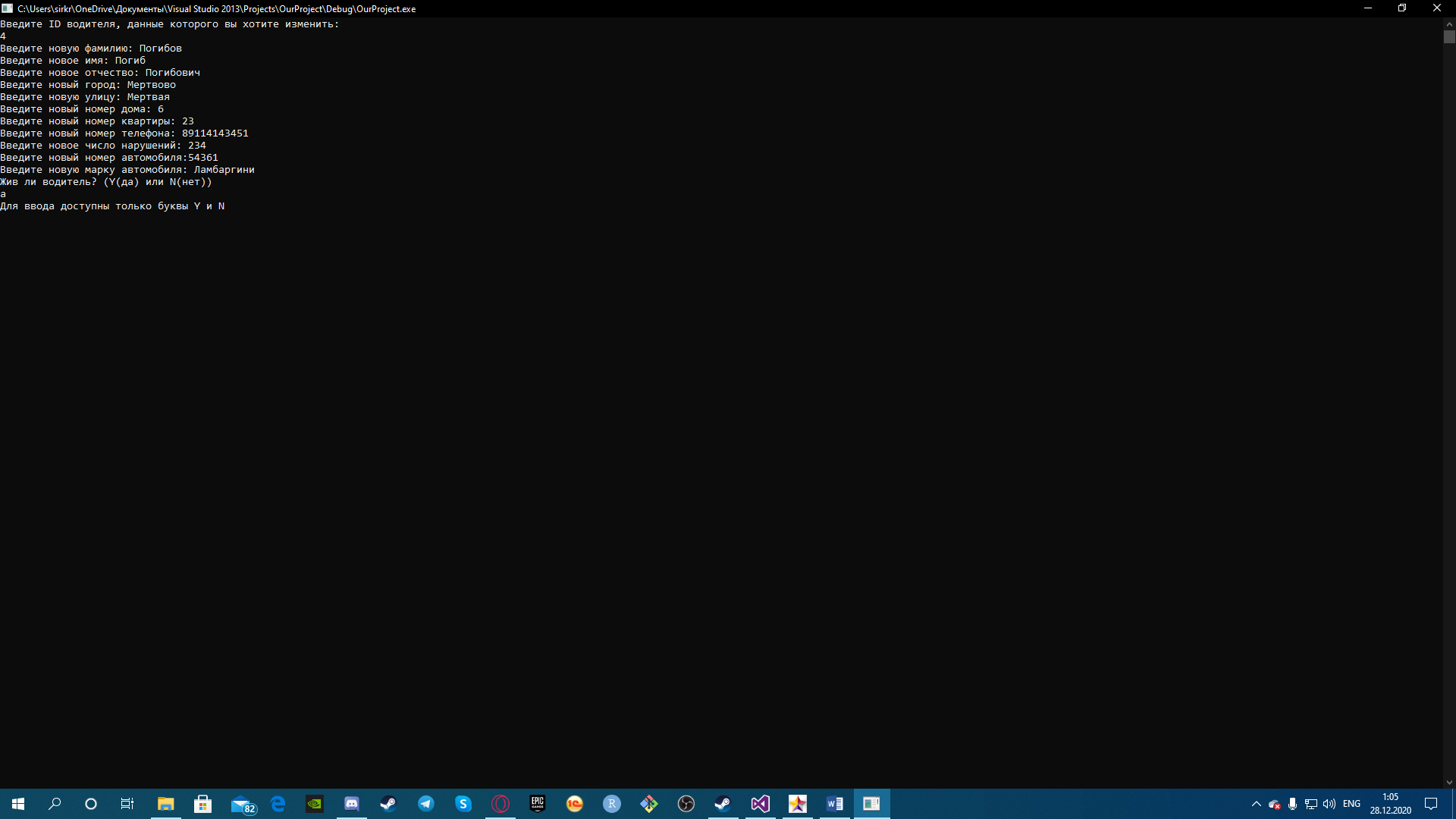
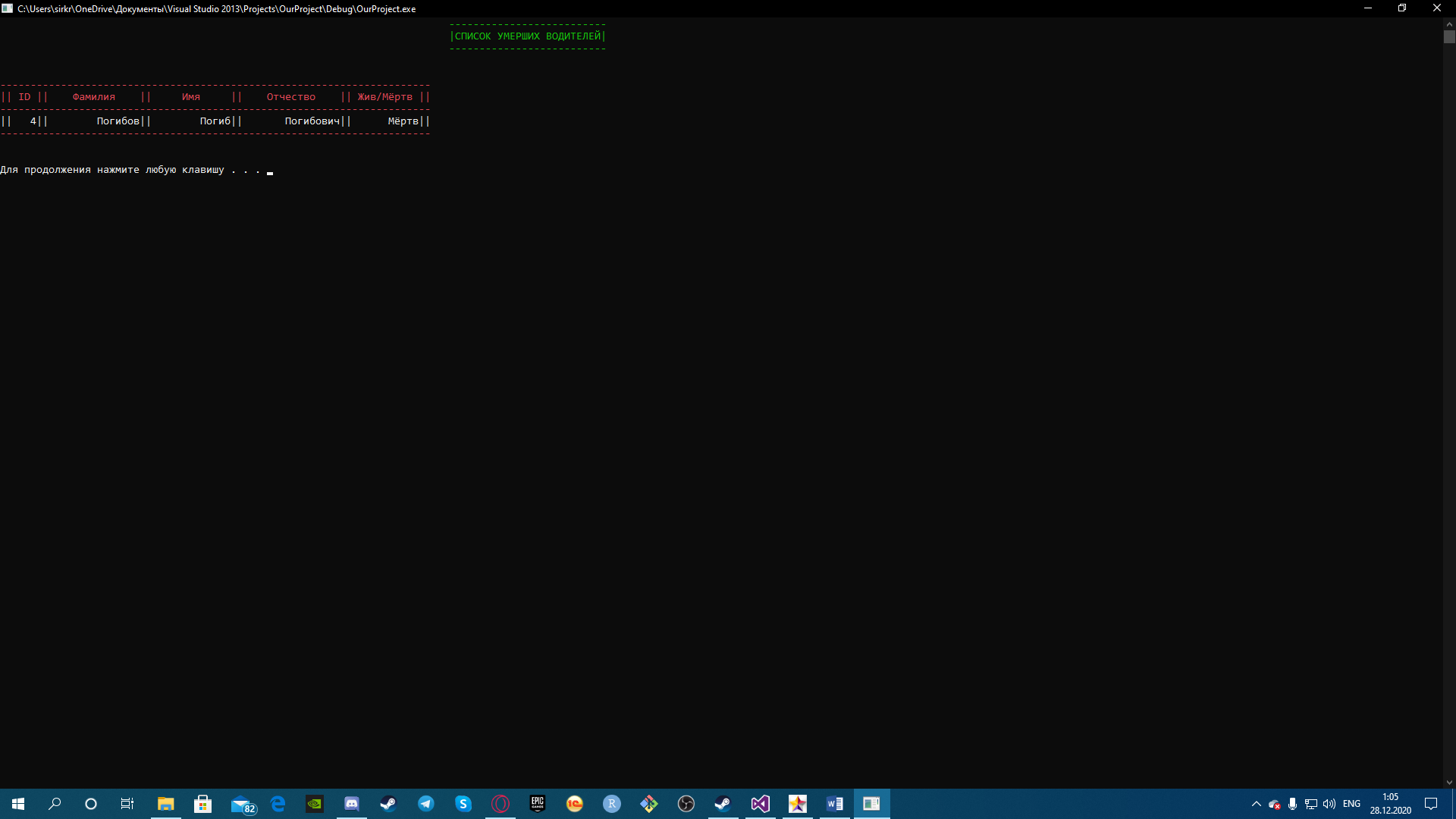


Рис. 13 – Режим редактирования водителя (в данном случае приведем пример с погибшим водителем)

При необходимости получить таблицу с умершими водителями Вам необходимо нажать горячую клавишу F3. После данного действия система выдаст вам данные о погибших водителях.

  
Рис. 14 – Вид на таблицу с умершими водителями

Если Вам необходимо вывести список водителей, которые нарушают правила (нарушителем считается водитель, который имеет 1 и более количество нарушений), то Вам следует нажать клавишу F4, которая произведет запрос в систему и Вы получите нужные данные.

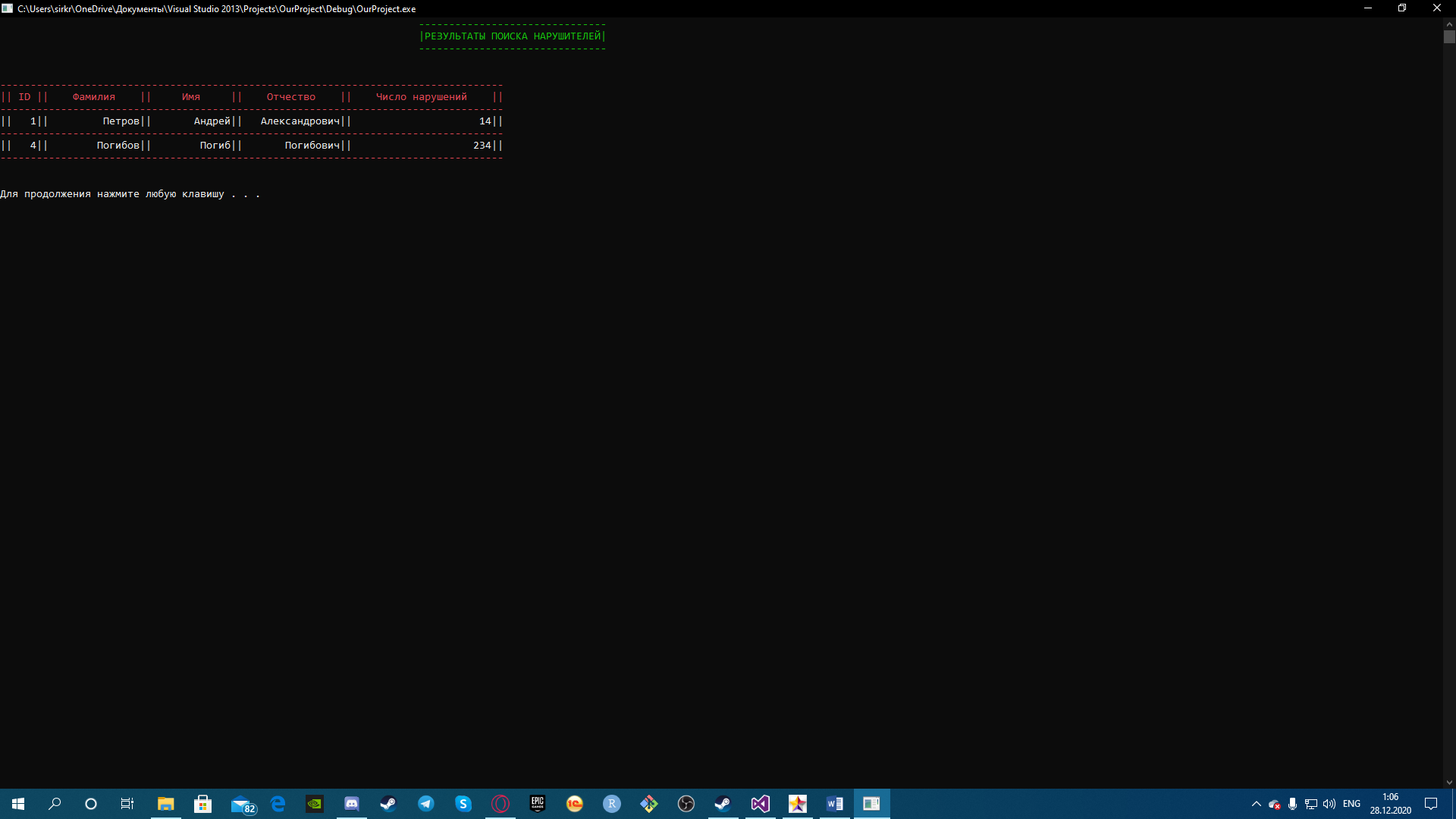


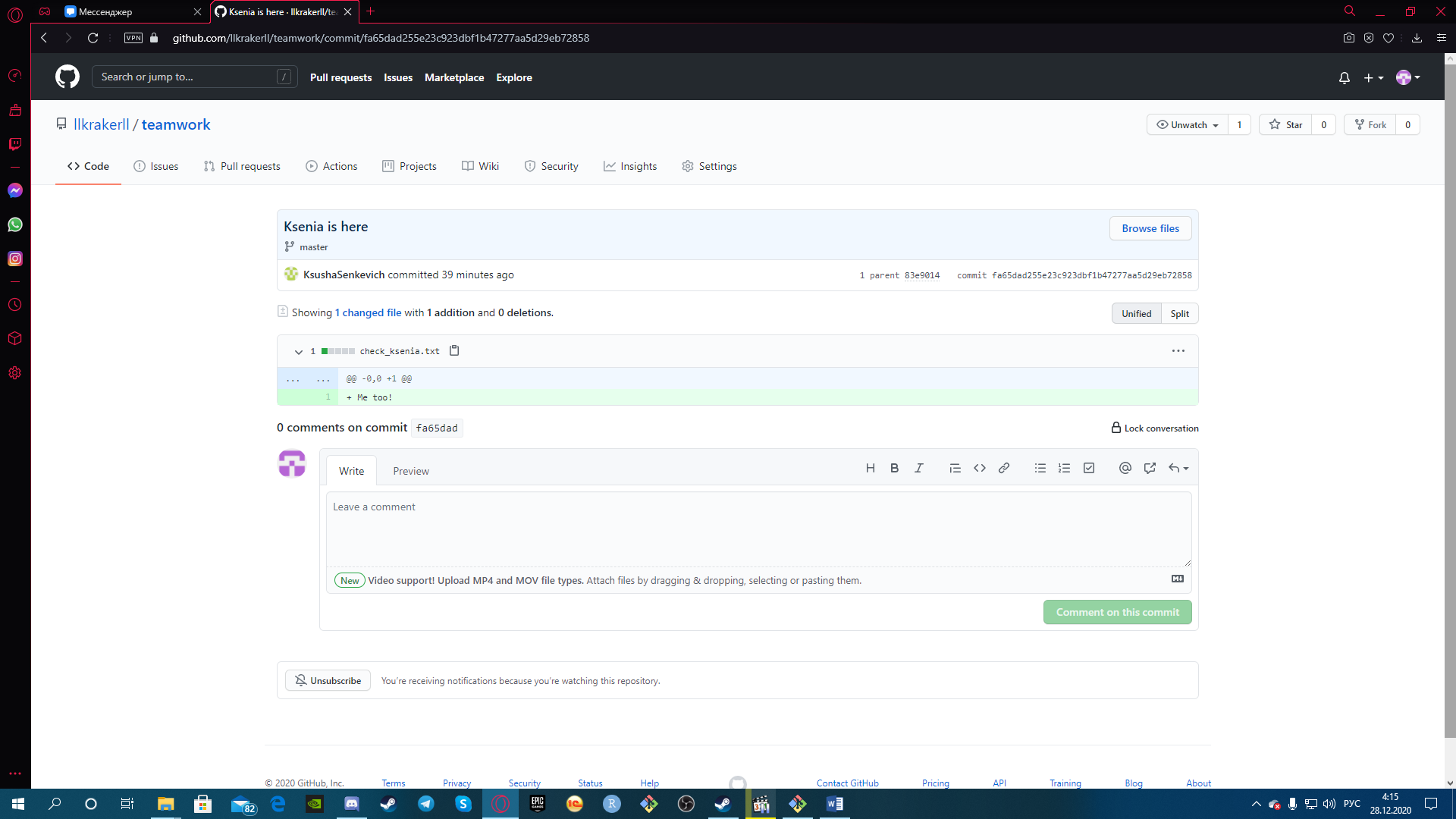
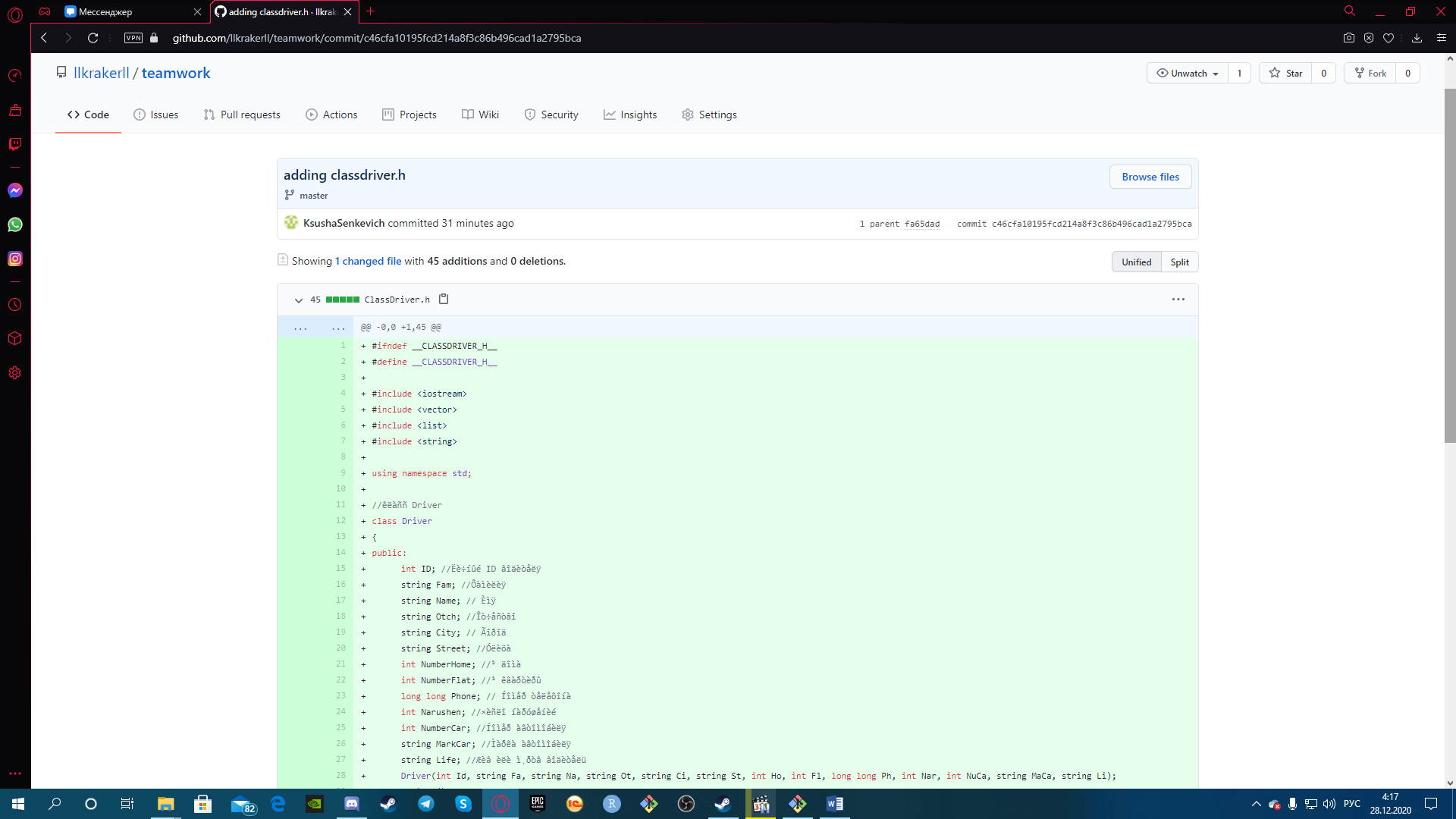
Рис. 15 – Вид на таблицу нарушителей

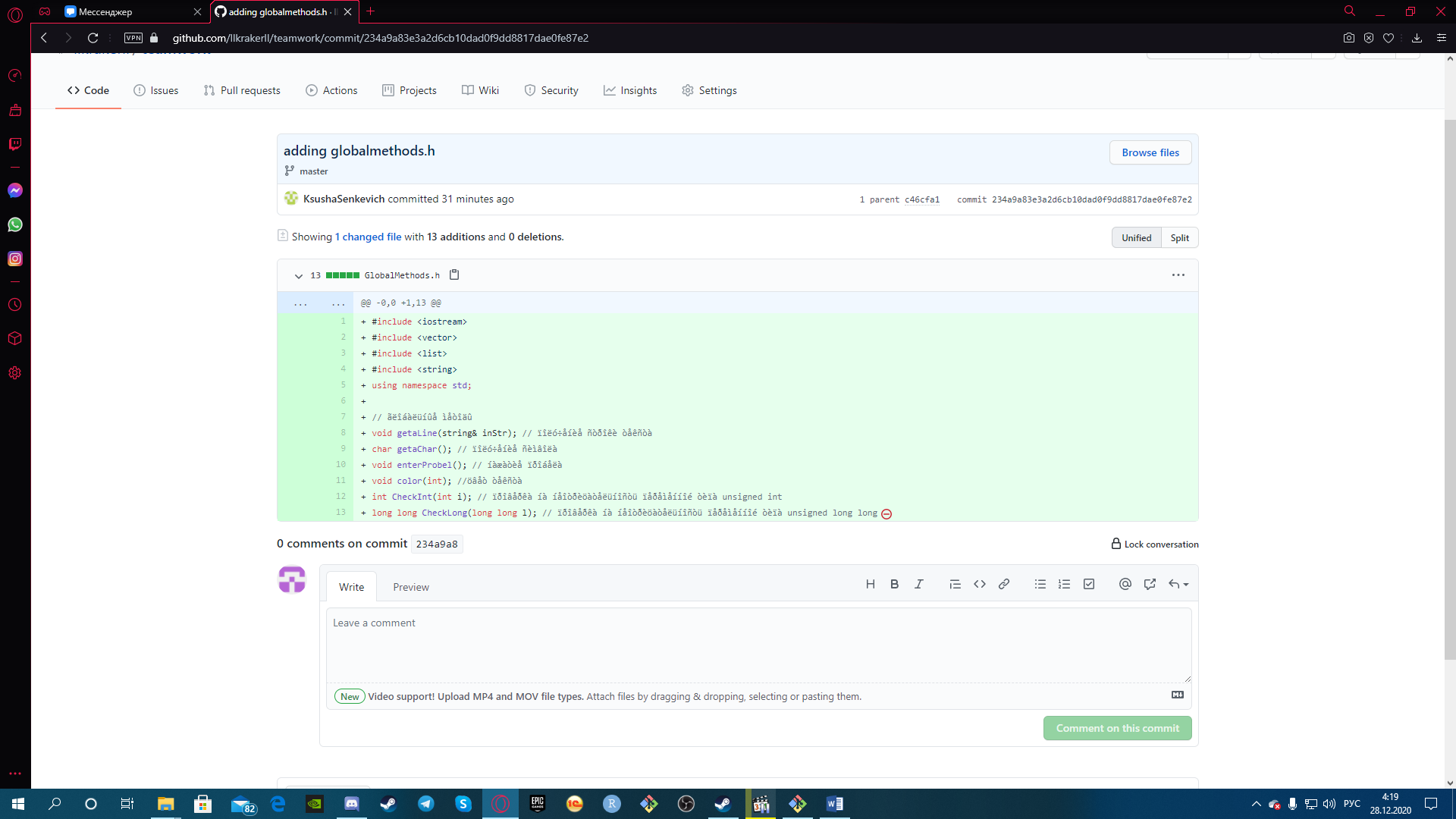
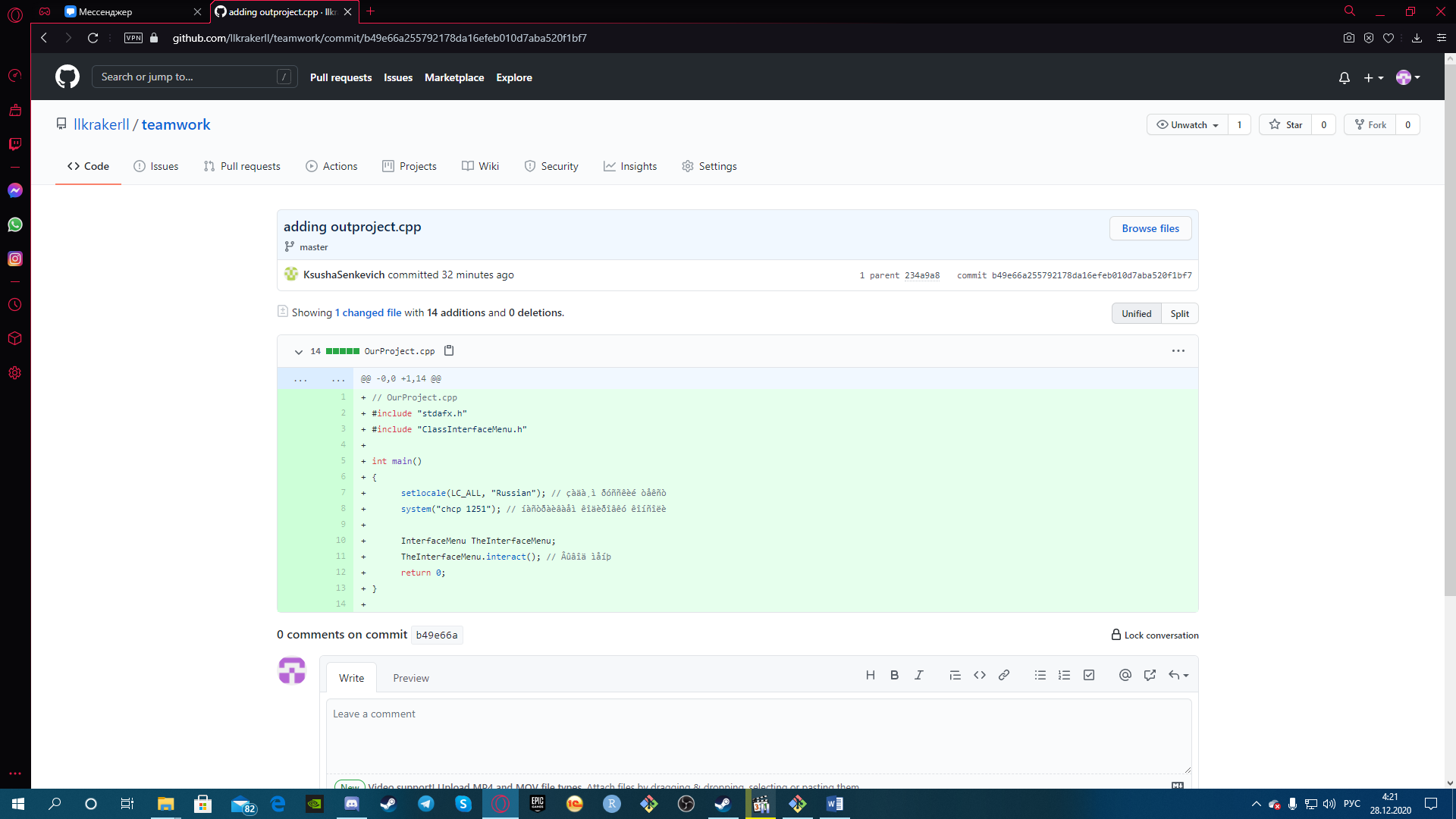
Если Вам необходимо закончить работу, то нажатие клавиши F5 приведет к тому, что система завершится и Вы выйдите из программы.

История проекта на GitHub

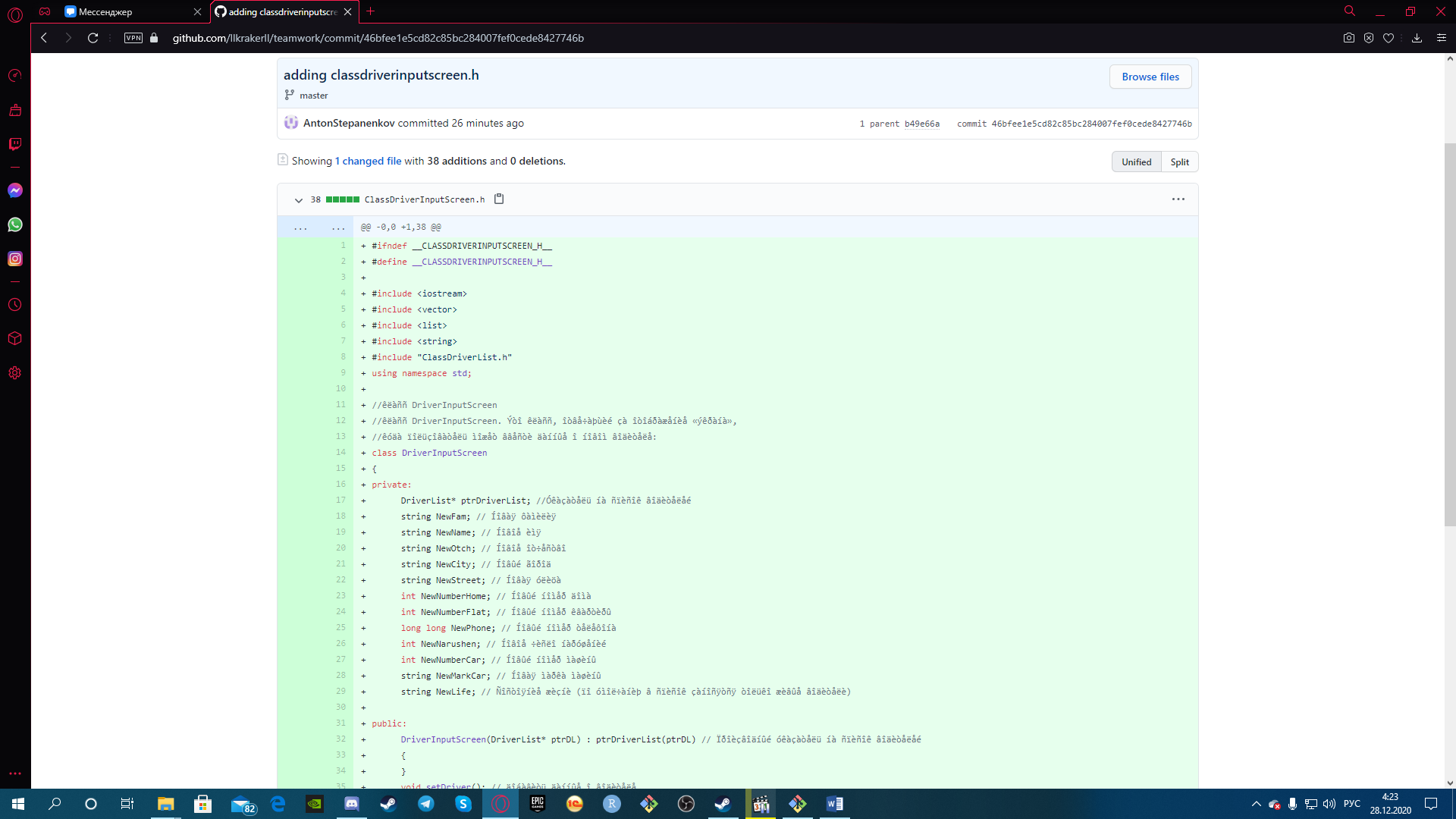
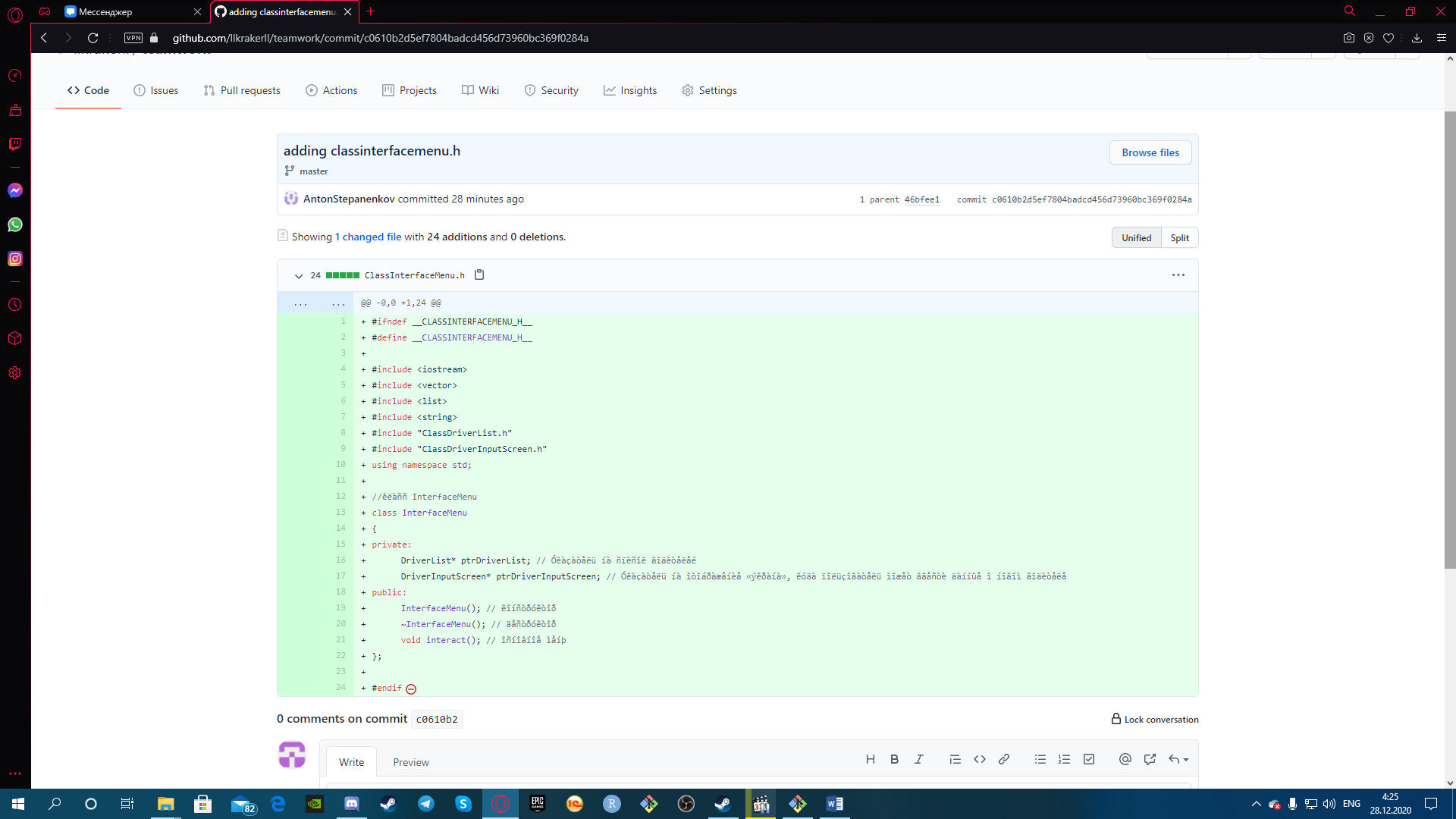
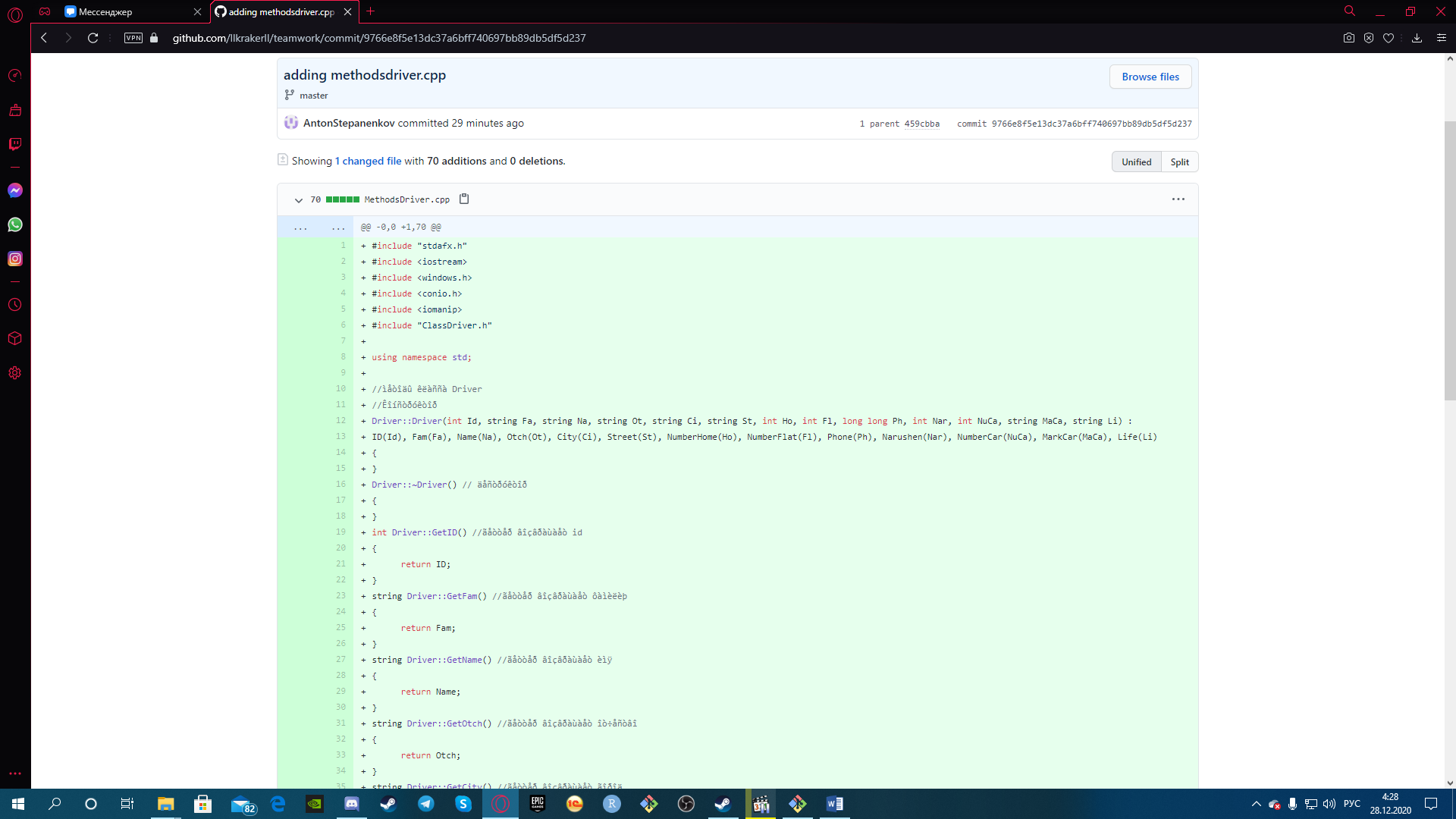
На наш совместный проект можно перейти по ссылке  
https://github.com/llkrakerll/teamwork.git

Сенкевич Ксения:

  
Рис.16 – Первый коммит. Проверка соединения с GitHub  
Рис.17 – Второй коммит. Добавление файла ClassDriver.h

  
Рис.18 – Третий коммит. Добавление файла GlobalMethods.h  
  
Рис.19 – Четвертый коммит. Добавление файла OurProject.cpp

Степаненков Антон:

  
Рис.20 – Первый коммит. Добавление файла ClassDriverInputScreen.h  
  
Рис.21 – Второй коммит. Добавление файла ClassInterfaceMenu.h  
  
Рис.22 – Третий коммит. Добавление файла InterfaceMenu.cpp  
  
Рис.23 – Четвертый коммит. Добавление файла MethodsDriver.cpp

Петров Андрей:

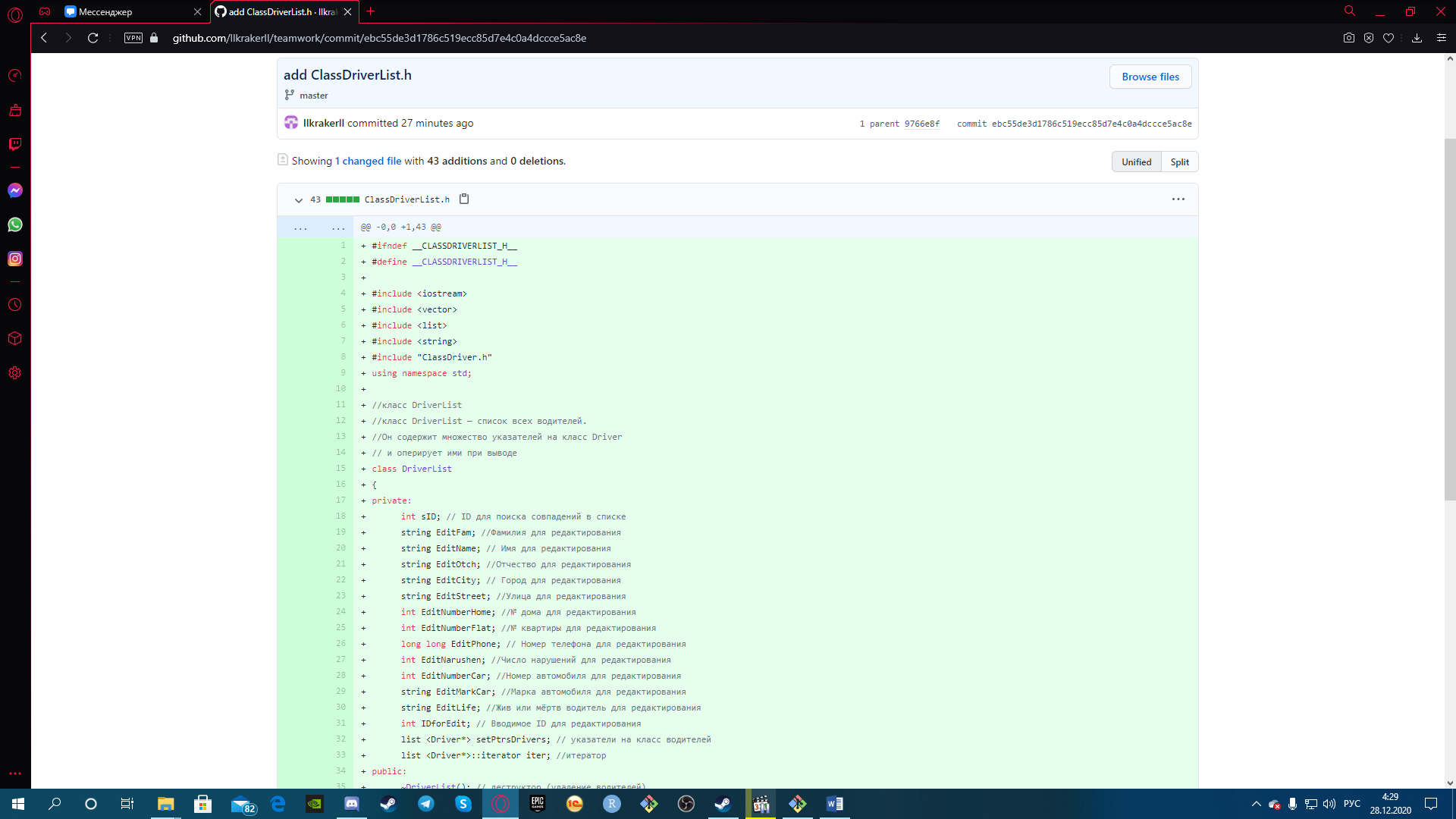
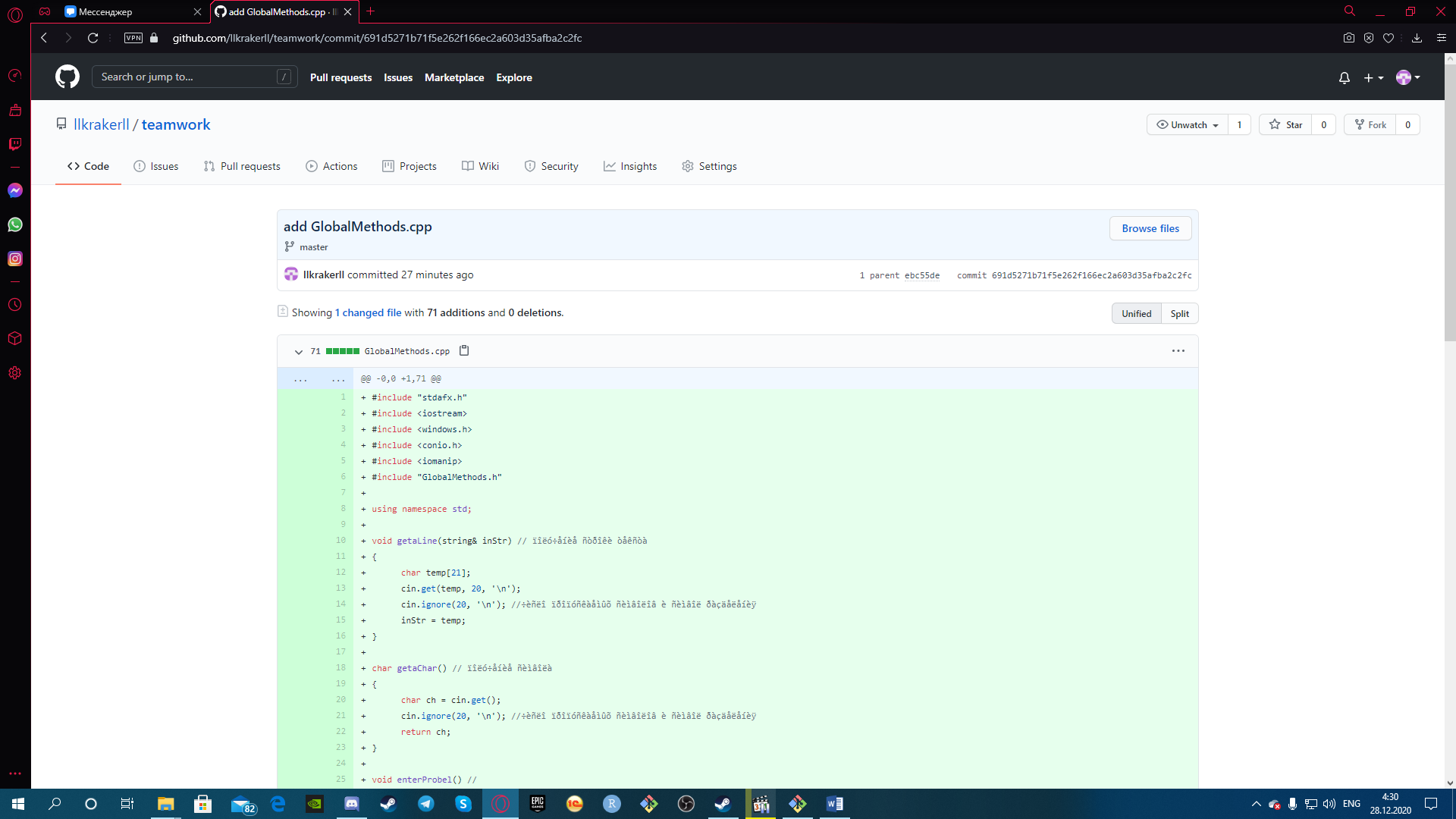
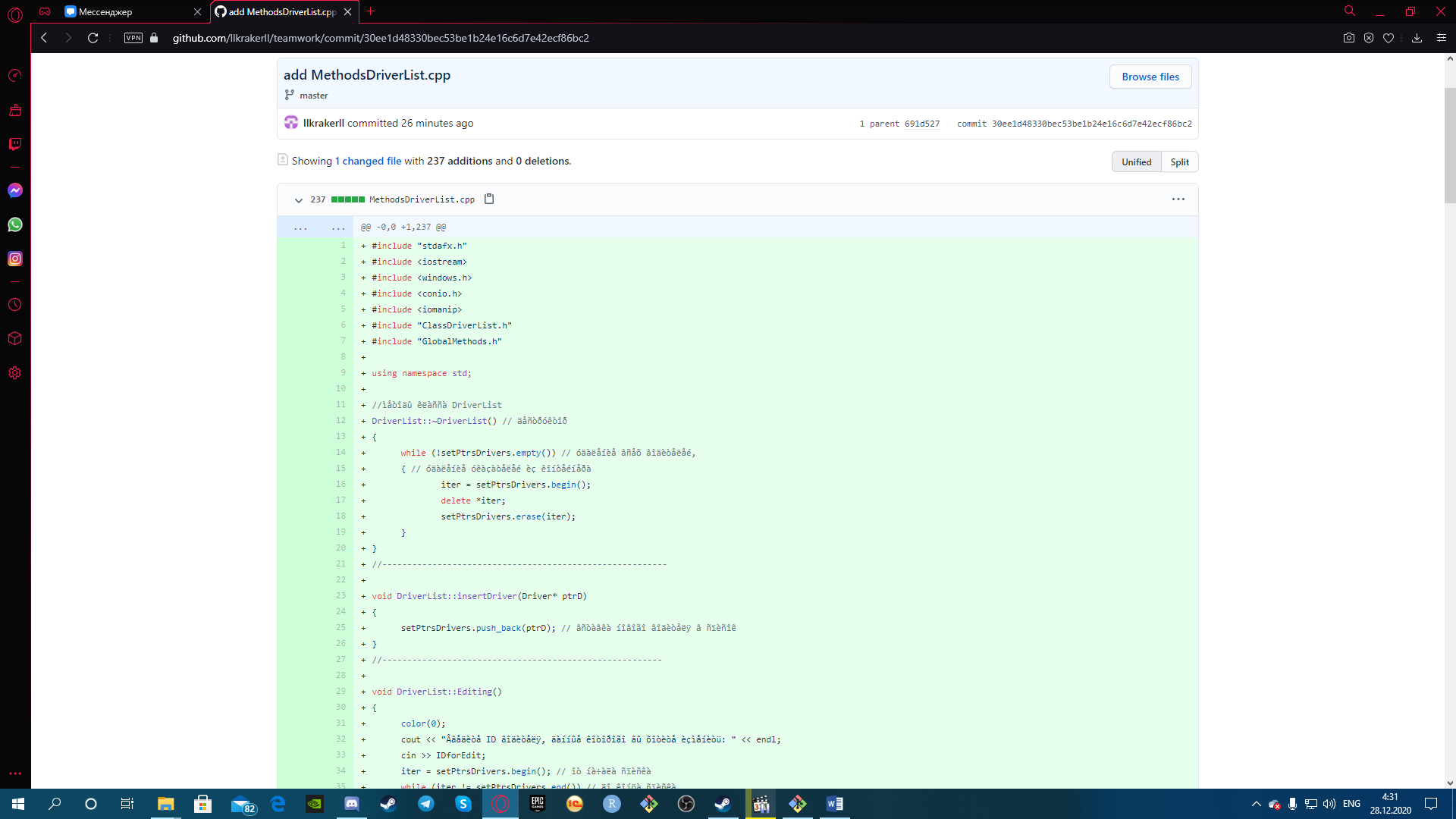
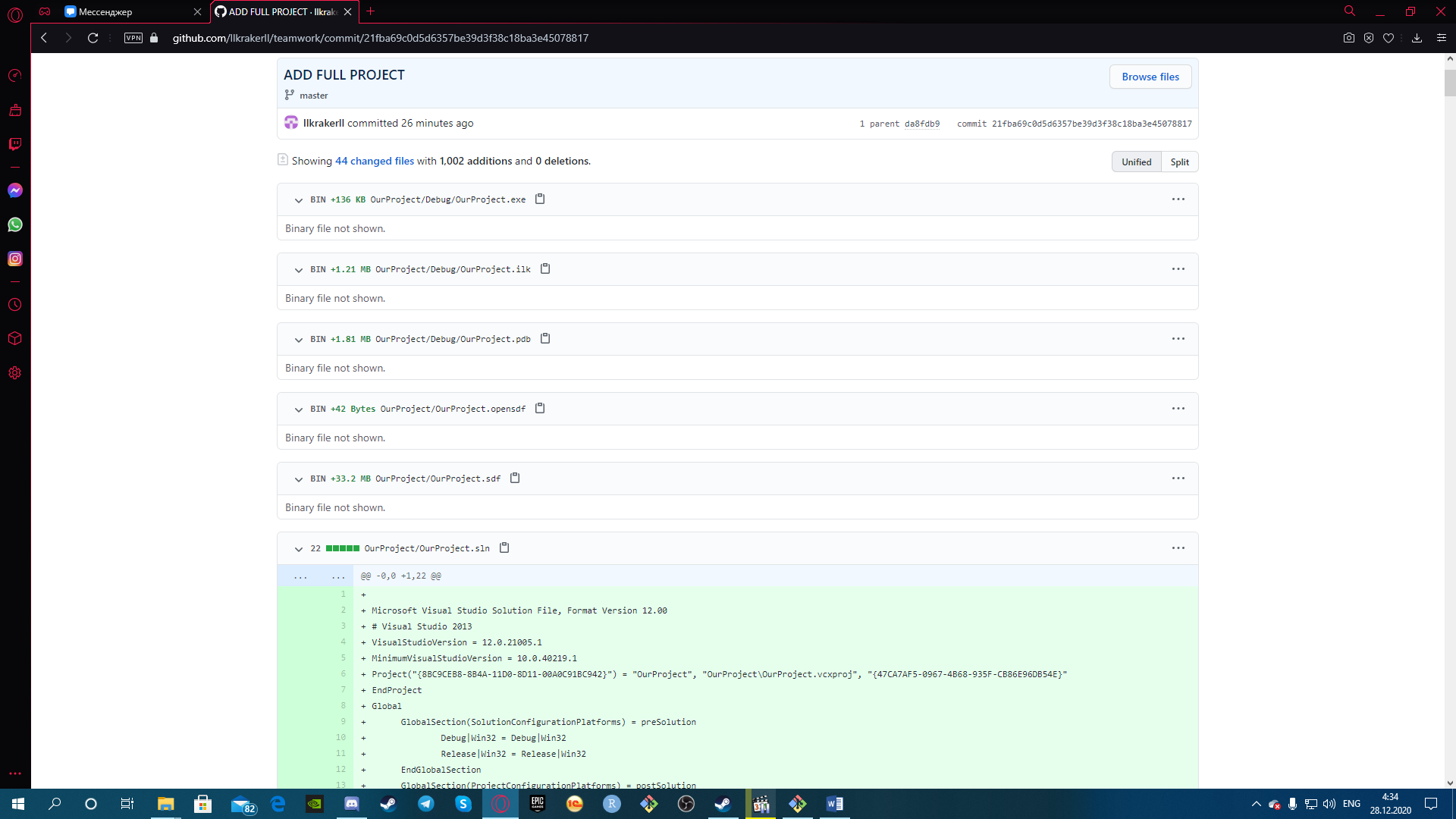
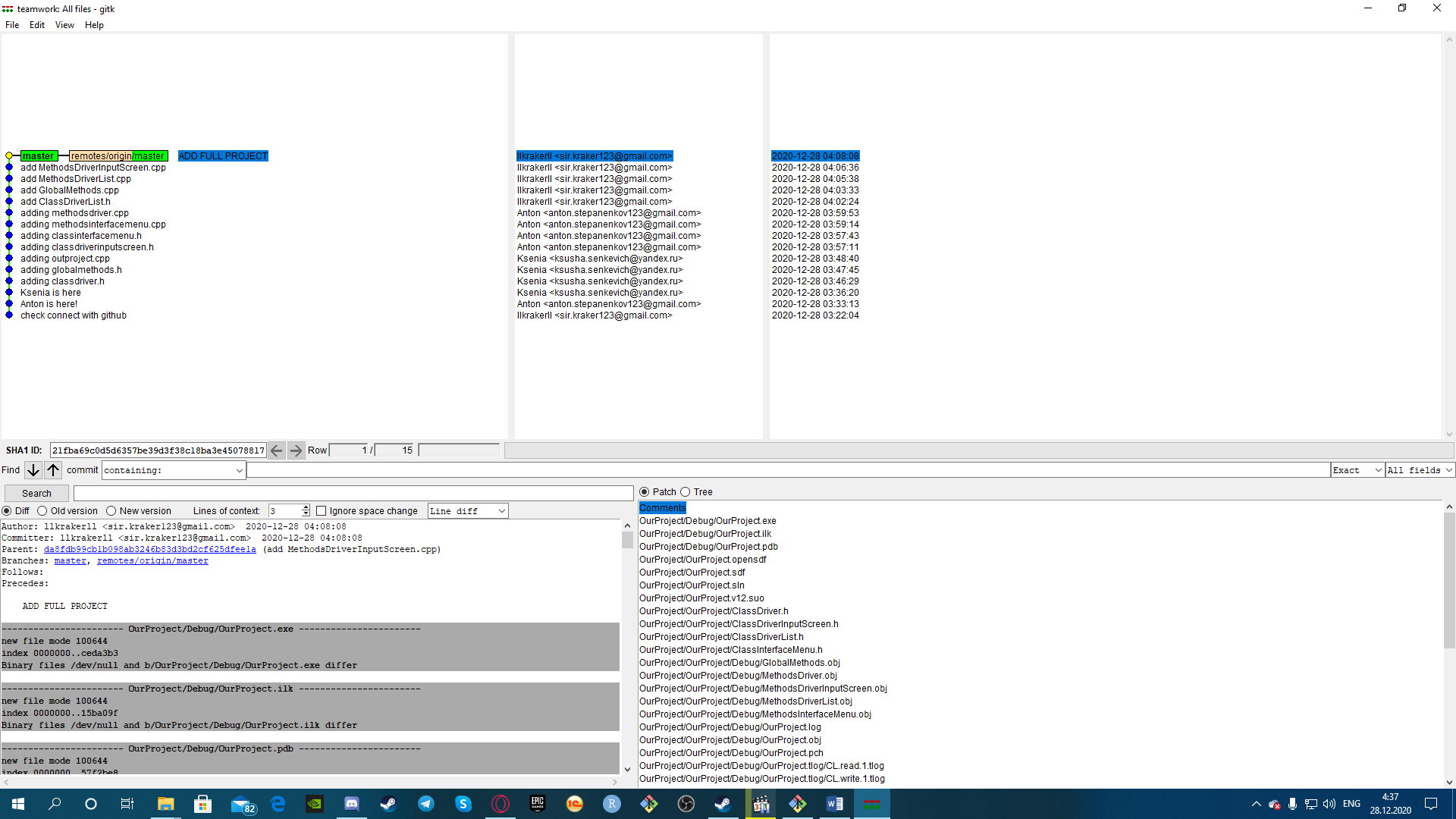
  
Рис.24 – Первый коммит. Добавление файла ClassDriverList.h  
  
Рис.25 – Второй коммит. Добавление файла GlobalMethods.cpp  
  
Рис.26 – Третий коммит. Добавление файла MethodsDriverList.cpp

  
Рис.27 – Четвертый коммит. Добавление файла MethodsDriverInputScreen.cpp  
  
Рис.28 - Пятый коммит. Добавление общей папки

  
Рис.29 История с gitk

Заключение

Наш проект был разработан в среде Microsoft Visual Studio 13 на языке C++. Нами была применена система контроля версий Git. В программе не наблюдается никаких зависаний и сбоев. Так же в проекте используется система раздельной компиляции. Очистка памяти реализована корректно с помощью ряда деструкторов. Все переменные используются и не существует лишних частей. Все конструкции являются необходимыми (ненужных и неиспользуемых алгоритмов в проекте не существует). В отчете нами были приложены диаграммы классов, диаграммы вариантов использования и диаграммы действий. Программа полностью достигла поставленной цели и выполняет все точно и быстро.