**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

Факультет фундаментальной и прикладной информатики

полное наименование кафедры

Кафедра программной инженерии  полное наименование кафедры

Направление подготовки (специальность) 09.03.04 Программная инженерия  шифр и название направления подготовки, специальности

**ОТЧЕТ**

о производственной практике (научно-исследовательской работе)

наименование вида и типа практики

на (в) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

наименование предприятия, организации, учреждения

студента 3 курса, группы ПО-61б\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

курса, группы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

фамилия, имя, отчество

Руководитель практики от Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_

предприятия, организации,

учреждения

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

должность, фамилия, и. о. подпись, дата

Руководитель практики от Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_

университета

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

должность, звание, степень

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
фамилия, и. о. подпись, дата

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата фамилия, и. о.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата фамилия, и. о.

Курск, 2019

# Реферат

Данный текстовый документ имеет объем 22 страниц, количество иллюстраций составляет - 4, таблиц - 0, приложений - 1. При работе над данной практической работой было использовано 13 источников литературы.

Перечень ключевых слов: интерфейс, цикл, переменная, матрица, программа, кнопка, таблица.

Выполняя задание по практике, требовалось разработать программу, которая анализировала бы предоставленный код на языке С#

Среда разработки программы: Borland C++ Builder 6.

# Содержание

Реферат 2

Содержание 3

Введение 4

1 Сведения о профессиональной организации 5

1.1 Профиль деятельности 5

2 Постановка задачи 6

2.1 Требования к выполнению 6

2.2 Теоретические сведения 6

2.3 Решение поставленной задачи 11

2.4 Интерфейс информационной системы 12

2.5 Тестирование информационной системы 12

Заключение 16

Список используемых источников 17

Приложение А 19

# Введение

Прохождение производственной практики осуществляется исходя из государственного стандарта, определяющего уровень базовой подготовки бакалавров в области программной инженерии.

Основные цели производственной практики: приобретение необходимых практических навыков в основных областях профессиональной деятельности.

Задачами прохождения практики являются:

* получение навыков практической деятельности;
* проведение научного исследования;
* закрепление в ходе практической работы знаний, полученных при изучении специальных дисциплин и дисциплин специализации;
* приобретение умений и навыков в тестировании и отладке программного обеспечения.
* сбор фактических, статических данных, для выполнений выпускной квалификационной работы.

Практика была пройдена на ООО «Дело 42», сферы деятельности которой: информационные технологии, системная интеграция, интернет.

# Сведения о профессиональной организации

## **1.1 Профиль деятельности**

EFSOL – динамично развивающаяся компания, которая с 2004 года предоставляет услуги системной интеграции – автоматизации учета на предприятиях на базе программного обеспечения 1С, а также широкий спектр ИТ-услуг на российском и международном рынке.

Учреждение осуществляет следующие основные виды деятельности:

* Автоматизация производства – комплексная автоматизация производственного предприятия;
* Автоматизация продаж - Комплексная автоматизация торгового предприятия;
* Облачный провайдер - Предоставление корпоративных облачных услуг;
* Автоматизация документооборота - Электронный документооборот Электронный архив;
* Абонентское обслуживание 1С - Полный спектр услуг по обслуживанию систем на базе программного обеспечения 1С;
* ИТ-аутсорсинг - Построение и поддержка ИТ-инфраструктуры предприятий.

# 2 Постановка задачи

## **2.1 Требования к выполнению**

Требуется автоматизировать проверку правильности написания кода на языке С#.

## **2.2 Теоретические сведения**

### **Интерфейсы**

***Интерфейс (interface)*** представляет собой не более чем просто именованный набор абстрактных членов. Абстрактные методы являются чистым протоколом, поскольку не имеют никакой стандартной реализации. Конкретные члены, определяемые интерфейсом, зависят от того, какое поведение моделируется с его помощью. Это действительно так. Интерфейс выражает поведение, которое данный класс или структура может избрать для поддержки. Более того, каждый класс (или структура) может поддерживать столько интерфейсов, сколько необходимо, и, следовательно, тем самым поддерживать множество поведений.

Нетрудно догадаться, что в библиотеках базовых классов .NET поставляются сотни предопределенных типов интерфейсов, которые реализуются в различных классах и структурах. Например, в состав ADO.NET входит множество поставщиков данных, которые позволяют взаимодействовать с определенной системой управления базами данных. Это означает, что в ADO.NET на выбор доступно множество объектов соединения (SqlConnection, OracleConnection, OdbcConnection и т.д.).

В интерфейсе ни у одного из методов не должно быть тела. Это означает, что в интерфейсе вообще не предоставляется никакой реализации. В нем указывается только, что именно следует делать, но не как это делать. Как только интерфейс будет определен, он может быть реализован в любом количестве классов. Кроме того, в одном классе может быть реализовано любое количество интерфейсов.

Для реализации интерфейса в классе должны быть предоставлены тела (т.е. конкретные реализации) методов, описанных в этом интерфейсе. Каждому классу предоставляется полная свобода для определения деталей своей собственной реализации интерфейса. Следовательно, один и тот же интерфейс может быть реализован в двух классах по-разному. Тем не менее в каждом из них должен поддерживаться один и тот же набор методов данного интерфейса. А в том коде, где известен такой интерфейс, могут использоваться объекты любого из этих двух классов, поскольку интерфейс для всех этих объектов остается одинаковым. Благодаря поддержке интерфейсов в C# может быть в полной мере реализован главный принцип полиморфизма: один интерфейс — множество методов.

Интерфейсы объявляются с помощью ключевого слова **interface**. Ниже приведена упрощенная форма объявления интерфейса:

*interface имя{*

*возвращаемый\_тип имя\_метода\_1 (список\_параметров);*

*возвращаемый\_тип имя\_метода\_2 (список\_параметров);*

*// ...*

*возвращаемый\_тип имя\_метода\_N (список\_параметров);*

*}*

где имя — это конкретное имя интерфейса. В объявлении методов интерфейса используются только их возвращаемый\_тип и сигнатура. Они, по существу, являются абстрактными методами. Как пояснялось выше, в интерфейсе не может быть никакой реализации. Поэтому все методы интерфейса должны быть реализованы в каждом классе, включающем в себя этот интерфейс. В самом же интерфейсе методы неявно считаются открытыми, поэтому доступ к ним не нужно указывать явно.

Помимо методов, в интерфейсах можно также указывать свойства, индексаторы и события. Интерфейсы не могут содержать члены данных. В них нельзя также определить конструкторы, деструкторы или операторные методы. Кроме того, ни один из членов интерфейса не может быть объявлен как static.

Как только интерфейс будет определен, он может быть реализован в одном или нескольких классах. Для реализации интерфейса достаточно указать его имя после имени класса, аналогично базовому классу. Ниже приведена общая форма реализации интерфейса в классе:

*class имя\_класса : имя\_интерфейса {*

*// тело класса*

*}*

где имя\_интерфейса — это конкретное имя реализуемого интерфейса. Если уж интерфейс реализуется в классе, то это должно быть сделано полностью. В частности, реализовать интерфейс выборочно и только по частям нельзя.

В классе допускается реализовывать несколько интерфейсов. В этом случае все реализуемые в классе интерфейсы указываются списком через запятую. В классе можно наследовать базовый класс и в тоже время реализовать один или более интерфейс. В таком случае имя базового класса должно быть указано перед списком интерфейсов, разделяемых запятой.

Методы, реализующие интерфейс, должны быть объявлены как public. Дело в том, что в самом интерфейсе эти методы неявно подразумеваются как открытые, поэтому их реализация также должна быть открытой. Кроме того, возвращаемый тип и сигнатура реализуемого метода должны точно соответствовать возвращаемому типу и сигнатуре, указанным в определении интерфейса.

### **Работа с таблицами**

Элемент управления DataGridView предоставляет настраиваемую таблицу для отображения данных. Класс DataGridView допускает настройку ячеек, строк, столбцов и границ с помощью использования свойств, таких как DefaultCellStyle, ColumnHeadersDefaultCellStyle, CellBorderStyle и GridColor.

Можно воспользоваться элементом управления DataGridView для отображения данных вместе с источником базовых данных или без него. Не указывая источник данных, можно создать столбцы и строки, содержащие данные, и добавить их непосредственно в объект DataGridView с помощью свойств Rows и Columns. Можно также использовать коллекцию Rows для доступа к объектам DataGridViewRow и свойство DataGridViewRow.Cells для непосредственного считывания или записи значения ячейки. Индексатор Item также обеспечивает прямой доступ к ячейкам.

В качестве альтернативы заполнению элемента управления вручную можно задать свойства DataSource и DataMember, чтобы привязать объект DataGridView к источнику данных и автоматически заполнить его данными. Дополнительные сведения см. в разделе Отображение данных с помощью элемента управления DataGridView в Windows Forms.

При работе с очень большими объемами данных можно задать для свойства VirtualMode значение true, чтобы отобразить подмножество доступных данных. Виртуальный режим требует реализацию кэша данных, из которого будет заполняться элемент управления DataGridView.

### **Анализ предметной области**

C# − это язык программирования, который сочетает в себе объектно-ориентированные и контекстно-ориентированные концепции. При этом он является прямым потомком языков программирования С, C+ +, Delphi, Java. Этот язык разрабатывался в 1998-2001 гг. группой инженеров под руководством Андерсa Хейлсбергa в компании Microsoft, как основной язык разработки приложений для платформы Microsoft .NET. В настоящее время язык C# имеет много модификаций, последняя – C# 6.0. Компилятор с языком C# входит в стандартную установку самой .NET, поэтому программы на нем можно создавать и компилировать даже без инструментальных средств, типа Visual Studio.

Преимущества языка С# состоят в следующем: а) он претендует на подлинную объектную ориентированность и призван практически реализовать компонентно - ориентированный подход, способствующий меньшей машинно-архитектурной зависимости программного кода, а также большей гибкости, переносимости и легкости повторного использования (фрагментов) программ; б) его изначальная ориентация направлена на безопасность кода; в) система типизации более унифицирована и максимально близкая по масштабу и гибкости к Common Type System, принятой в Microsoft.NET; г) расширена поддержка событийно-ориентированного программирования; д) данный язык C# является как бы «родным» для создания приложений в среде Microsoft.NET, поскольку тесно и эффективно интегрирован с ней; е) язык C# не просто интегрированная сумма достоинств современных языков программирования (Java, C++, Visual Basic и др.), но и языков программирования нового поколения.

Вместе с тем языку программирования C# свойственны и недостатки. Во-первых, он имеет около 75% сложных синтаксических возможностей аналогичных Java, 10% подобных языку C++, а 5%, взятых из языка Visual Basic. В тоже время объем «свежих» концептуальных идей в языке C# составляет около 10% от общего объема конструкций языка. Поэтому утверждение, что язык C# является чисто объектным, допускает неоднозначную интерпретацию. К тому же, компилятор и среда разработки программного обеспечения, поддерживающие язык C#, обладают невысокой производительностью в виду того, что код программы на языке C# компилируется и выполняется примерно в 100 раз медленнее, чем тот же код на языке C++. Производительность же программ на языке C# сравнима с тем же показателем для языка Java. Следовательно, программы на языке C# не могут функционировать под управлением альтернативных операционных систем. Поэтому ведутся работы по обеспечению совместимости с ОС Linux и FreeBSD семейства UNIX, но эти отдельные недостатки носят скорее субъективный, локальный и временный, а не постоянный характер. Данный вывод подтверждает по версии TIOBE ProgrammingCommunityIndex популярность в мировом масштабе рейтинг языка C# в декабре 2013 и 2014 гг. среди применяемых программистами языков занимал 5-е место с удельным весом 4,33%, а в июле 2015 и 2016 гг.- 5 и 4 место соответственно с удельным весом в 3,92%.

Таким образом, язык программирования C# - «строго типизированный объектно- ориентированный язык, чей код внешне похож на С++ (и Java) и относится он к семье языков с С- подобным синтаксисом, из них его синтаксис также наиболее близок к С++ и Java, (хотя в некоторых отношениях отличается), но имеет строгую статическую типизацию, поддерживает полиморфизм, перегрузку операторов, указатели на функции-члены классов, атрибуты, события, свойства, исключения, комментарии в формате XML.

Проект является актуальным для разработчиков ПО на языке программирования C#, так как представляет собой программу, которая производит анализ и отбор ошибочных слов кода.

## **2.3 Решение поставленной задачи**

Для решения поставленной задачи, были выполнены следующие пункты:

* была создана кнопка, при нажатии на которую заполняется словарь ключевых слов языка С# из текстового файла;
* были созданы поля для ввода неправленного кода и вывода исправленного;
* было создано поле для вывода распознанных слов неправленного кода;

Программный код информационной системы приведен в Приложении А.

## **2.4 Интерфейс информационной системы**

На рисунке 1 продемонстрирован интерфейс программы

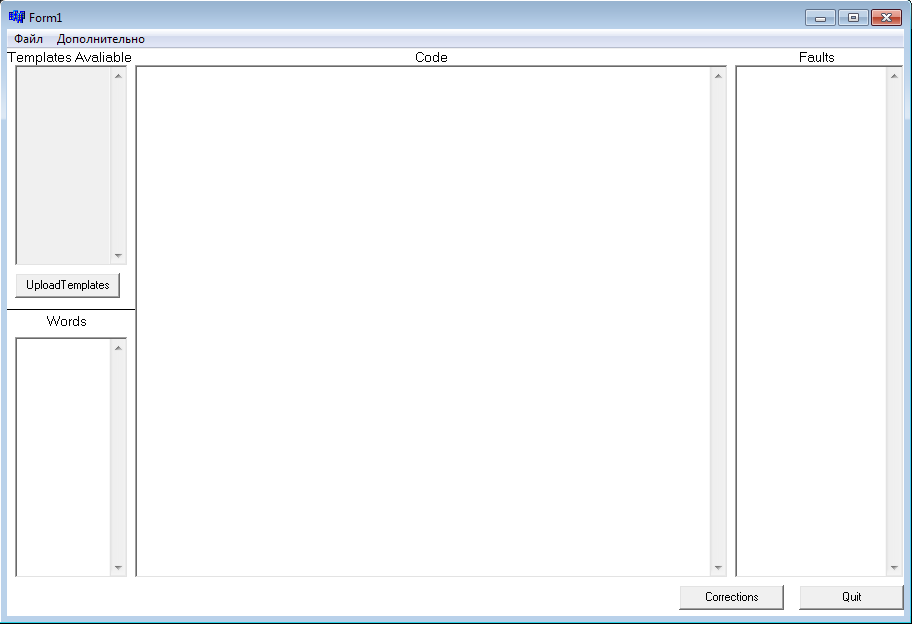


Рисунок 1 - Интерфейс программы

## **2.5 Тестирование информационной системы**

1. На рисунках 2, 3 и 4 проиллюстрированы нажатие кнопки «UploadTemplates» и кнопки «Corrections» с заполненным полeм «Templates Avaliabale», с синтаксически правильным, и затем неправильным кодом.

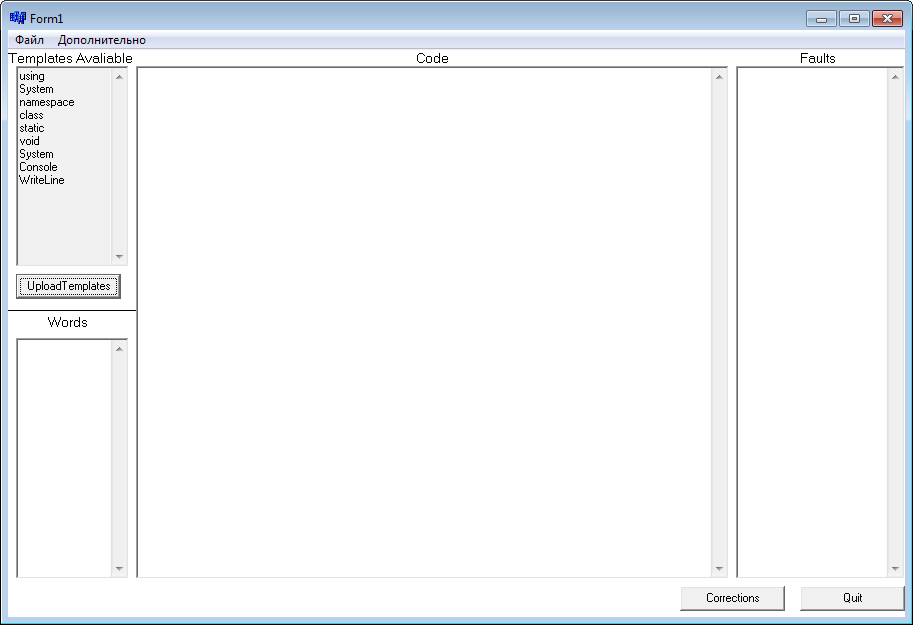


Рисунок 2 – Загрузка файла templates.txt

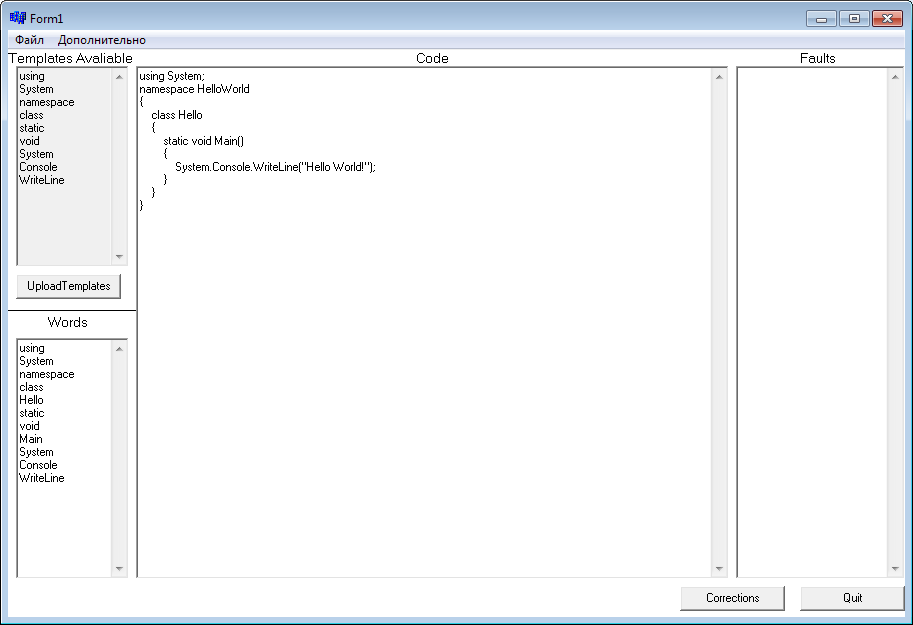


Рисунок 3 – Проверка правильного кода на ошибки

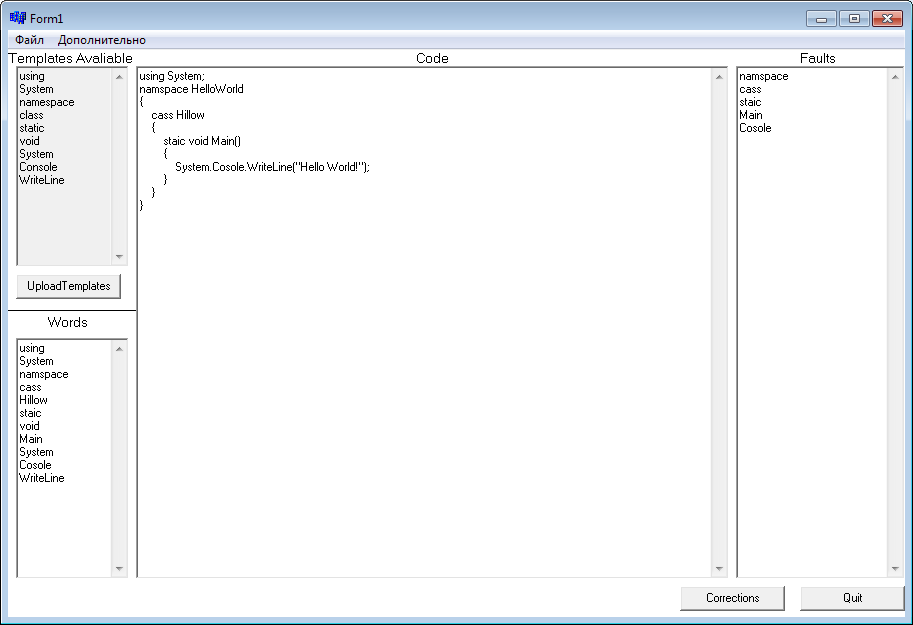


Рисунок 4 – Проверка кода с ошибками в ключевых словах

# Заключение

За время практики были выполнено задание, в ходе которого требовалось автоматизировать принятие решения в условиях неопределенности по классическим и производным критериям.

Производственная практика помогла не только лучше разобраться в возможностях, предоставляемых компьютерной техникой и современным программным обеспечением, но и получить навыки работы с пользователями и овладеть различными навыками работы оператора ПК.

Были получены разнообразные практические навыки, а также закреплена теоретическая база в соответствии с изученными дисциплинами 3 курса направления подготовки 09.03.04 «Программная инженерия»

# Список используемых источников

1. Агуров, Павел C#. Сборник рецептов [Текст] / Павел Агуров. - М.: "БХВ-Петербург", 2012. - 432 c.

2. Албахари, Джозеф C# 3.0. Справочник [Текст]/ Джозеф Албахари , Бен Албахари. - М.: БХВ-Петербург, 2013. - 944 c.

* + 1. Прайс, Джейсон Visual C# 2.0. Полное руководство [Текст]/ Джейсон Прайс, Майк Гандэрлой. - М.: Век +, Корона-Век, Энтроп, 2010. - 736 c.
    2. <https://docs.microsoft.com/ru-ru/dotnet/csharp/> [Электронный ресурс] от 10.07.2019
    3. Грей, Джо Professional XML for .NET Developers [Текст] / Джо Грей. - М: Лори, 2015. - 642 с.
    4. Рихтер, Джеффри CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# [Текст] / Джеффри Рихтер. - М.: Питер, 2017. - 896 с.
    5. Трей, Нэш C# 2010. Ускоренный курс для профессионалов [Текст] / Нэш Трей. - М: Вильямс, 2010. - 592 с.
    6. Гриффитс, Иэн Программирование на C# 5.0 [Текст] / Йен Гриффитс. - М: Эксмо, 2014. - 1136 с.
    7. Скит, Джон C# для профессионалов. Тонкости программирования [Текст] / Джон Скит. - М: Вильямс, 2017. - 608 с.
    8. <http://progopedia.ru/language/csharp/> [Электронный ресурс] от 10.07.2019
    9. <https://mva.microsoft.com/en-us/training-courses/c-fundamentals-for-absolute-beginners-16169> [Электронный ресурс] от 10.07.2019
    10. <https://dotnetcademy.net/CSharp/Beginner> [Электронный ресурс] от 10.07.2019
    11. <http://csharp.net-tutorials.com/> [Электронный ресурс] от 10.07.2019

# Приложение А

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

#include <vcl.h>

#pragma hdrstop

#include <string.h>

#include "Unit1.h"

#include <fstream.h>

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

struct tlist

{ char word;

tlist\* next;

}\*p,\*head;

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

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

Memo1->Lines->Clear();

Memo2->Lines->Clear();

Memo3->Lines->Clear();

Memo4->Lines->Clear();

}

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

void \_\_fastcall TForm1::AnalyzeClick(TObject \*Sender)

{

ifstream tm;

tm.open("templates.txt");

char s[10];

if(!tm)

{ShowMessage("Файл пуст или отсутствует");

return;}

else

{

for(int i=1;!tm.eof();i++)

{

tm >> s;

Memo1->Lines->Add(s);

}

}}

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

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

AnsiString c="",s="",rigt="",falt;

for(int q=0;q<=Memo2->Lines->Count;q++)

{

s+=(AnsiString)Memo2->Lines->Strings[q]+' ';

}

for(int i=1;i<=s.Length();i++)

{

if(s[i]>='a'&&s[i]<='z'||s[i]>='A'&&s[i]<='Z')

{

c+=s[i];

}

else

{

if(c.Length()!=1){Memo3->Lines->Add(c);}

c="";

}

for(int j=0;j<Memo3->Lines->Count;j++){if(Memo3->Lines->Strings[j]==""){Memo3->Lines->Delete(j);}}

}

for(int r=0;r<Memo3->Lines->Count;r++)

{

for(int t=0;t<Memo1->Lines->Count;t++)

{

if(Memo3->Lines->Strings[r]==Memo1->Lines->Strings[t])

{

rigt=Memo3->Lines->Strings[r];

}

else

{

falt=Memo3->Lines->Strings[r];

}

}

if(rigt=="")

{

Memo4->Lines->Add(falt);

}

falt="";rigt="";

}

}

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

void \_\_fastcall TForm1::N2Click(TObject \*Sender)

{Close();}

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

void \_\_fastcall TForm1::N3Click(TObject \*Sender)

{ShowMessage("Здравствуйте:D");}

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

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{Close();}

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