1. Ministry of Education and Science of the Russian Federation
2. Peter the Great St. Petersburg Polytechnic University
3. —
4. Institute of Applied Mathematics and Mechanics.
5. **Department of Information Security of Computer Systems**
6. **Laboratory work №3**

**CREATING EXTENSION FOR MICROSOFT VISUAL STUDIO**

on the discipline " Object-oriented programming"

1. Completed
2. a student group 33636/1 A.V. Malinko
3. Teacher A.Y. Chernov
4. Saint Petersburg
5. 2018
6. Objective

Familiarity with the Microsoft Visual Studio Object Model Automation, the study of the basic principles of its organization, the development of skills plug-in for Microsoft Visual Studio in C #, practical application objects and events of the automation model to extend the functionality IDE, mastering the basics of the regular expression mechanism.

Theoretical information

There are several ways to extend the functionality of Microsoft Visual. Studio. But the most common is the integration of independent modules of the Extension Package type (**VSPackage**), for the management of which Visual Studio Automation Object Model is used. AT Visual Studio standard programming language support modules (for example, Visual C ++ or Visual C #) are implemented as extensions

VSPackage.

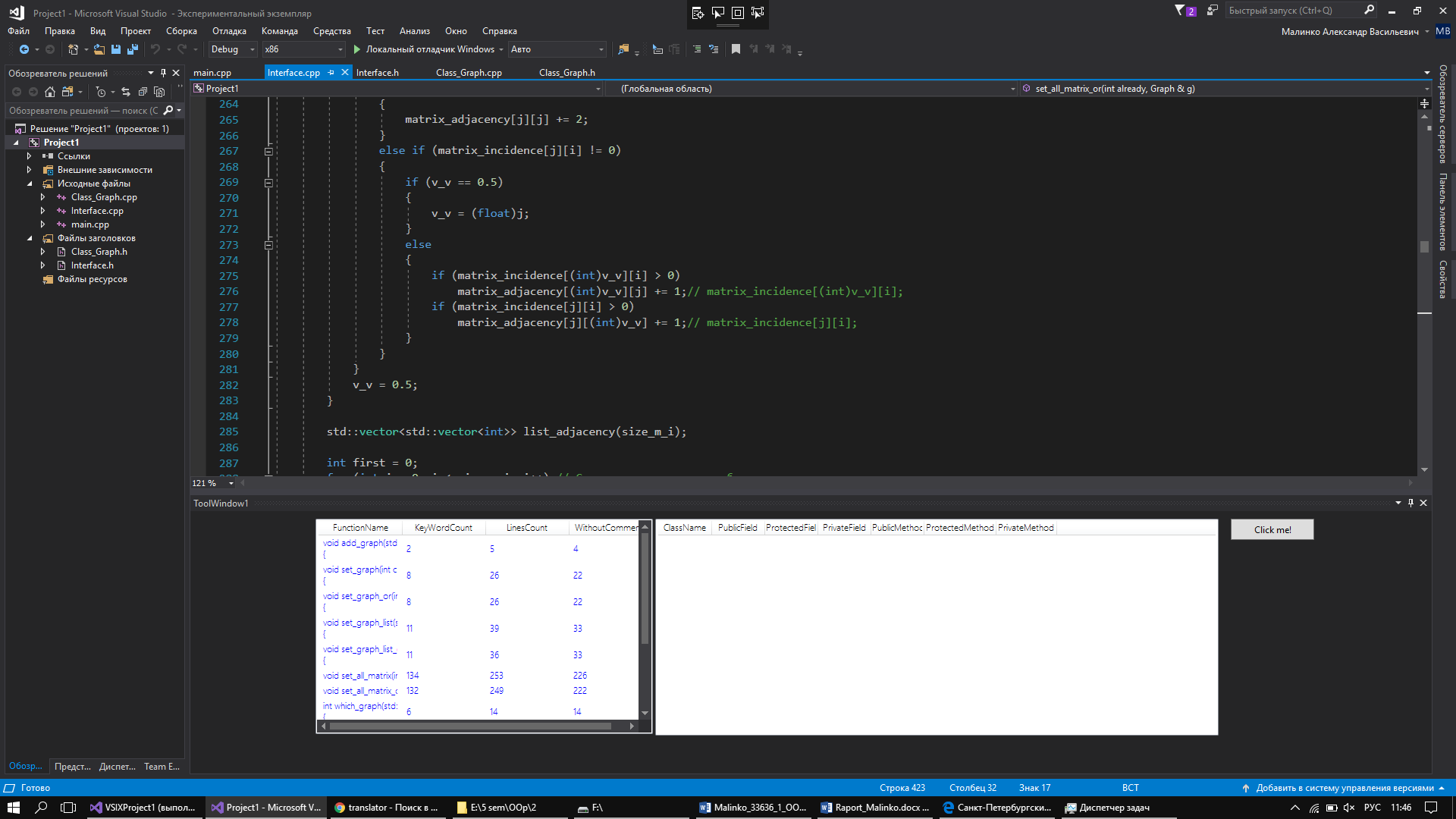
**Regular expressions** are a formal language for searching and implementing manipulations with underlays in the text, based on the use of metacharacters. For the search, a sample string consisting of characters and metacharacters is used, which specifies the search rule. For text manipulations, the replacement string is additionally specified, which can also contain special characters.

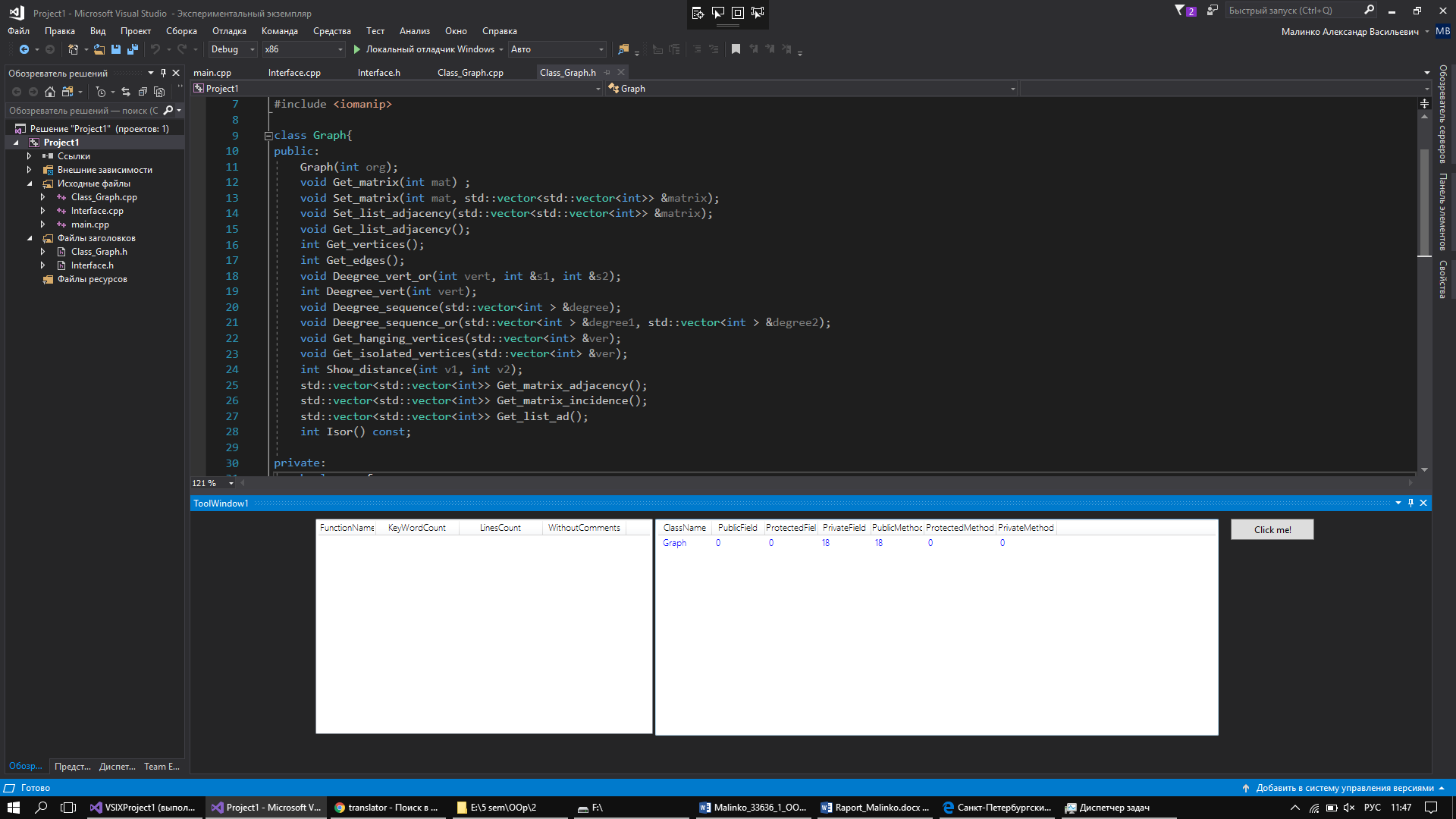
**Working process**

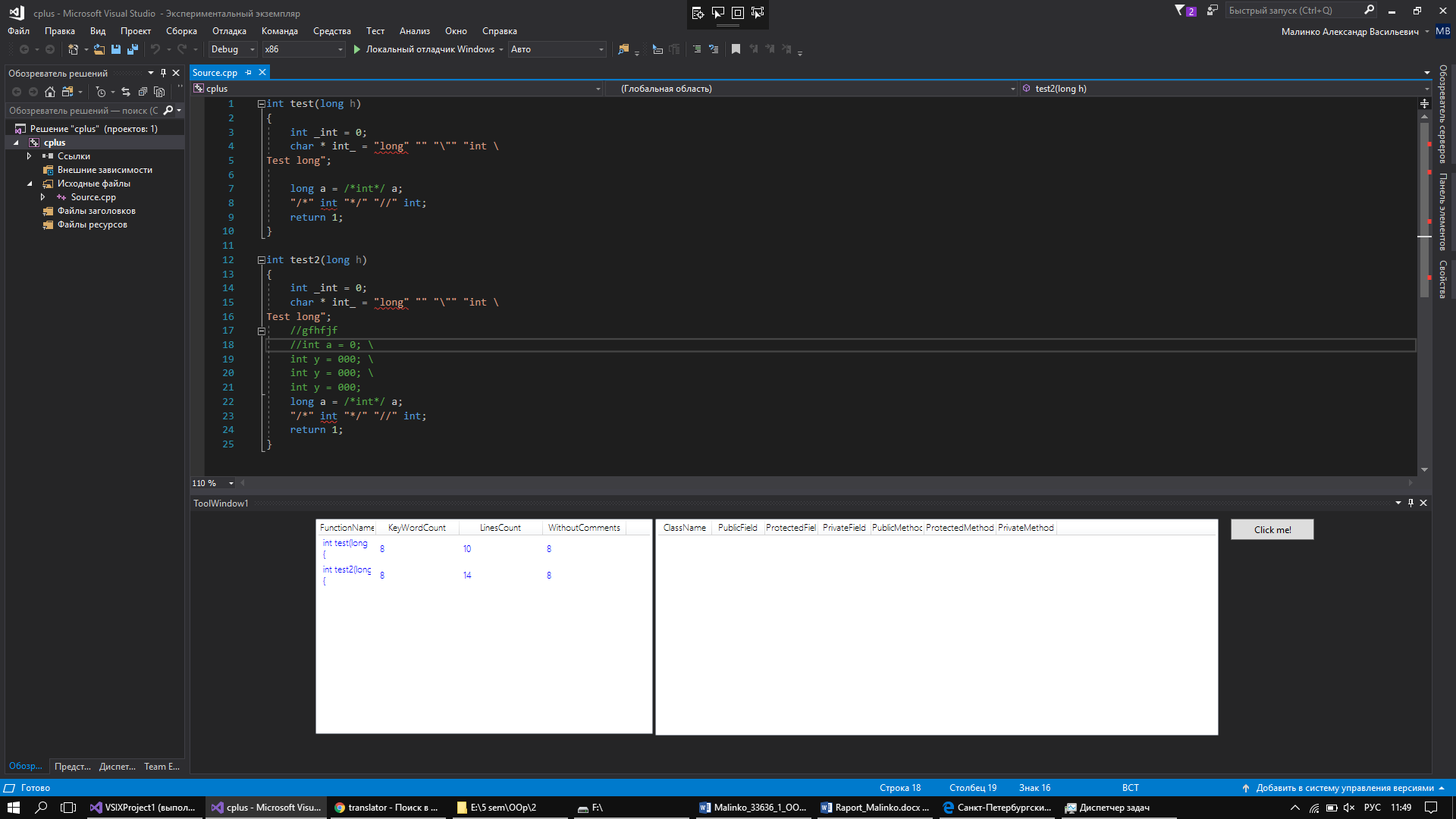
In the course of the work, an extension was created, which is an additional window that the user can move to a convenient place. Inside the window there are two tables and a button, which, when clicked, analyzes the code. In the code of an open project, all functions are distinguished in which all are highlighted: keywords (different for different languages), the number of lines of code, and the number of non-empty lines. For the program to work correctly, all lines and comments are removed from the code first. This is implemented using regular expressions.

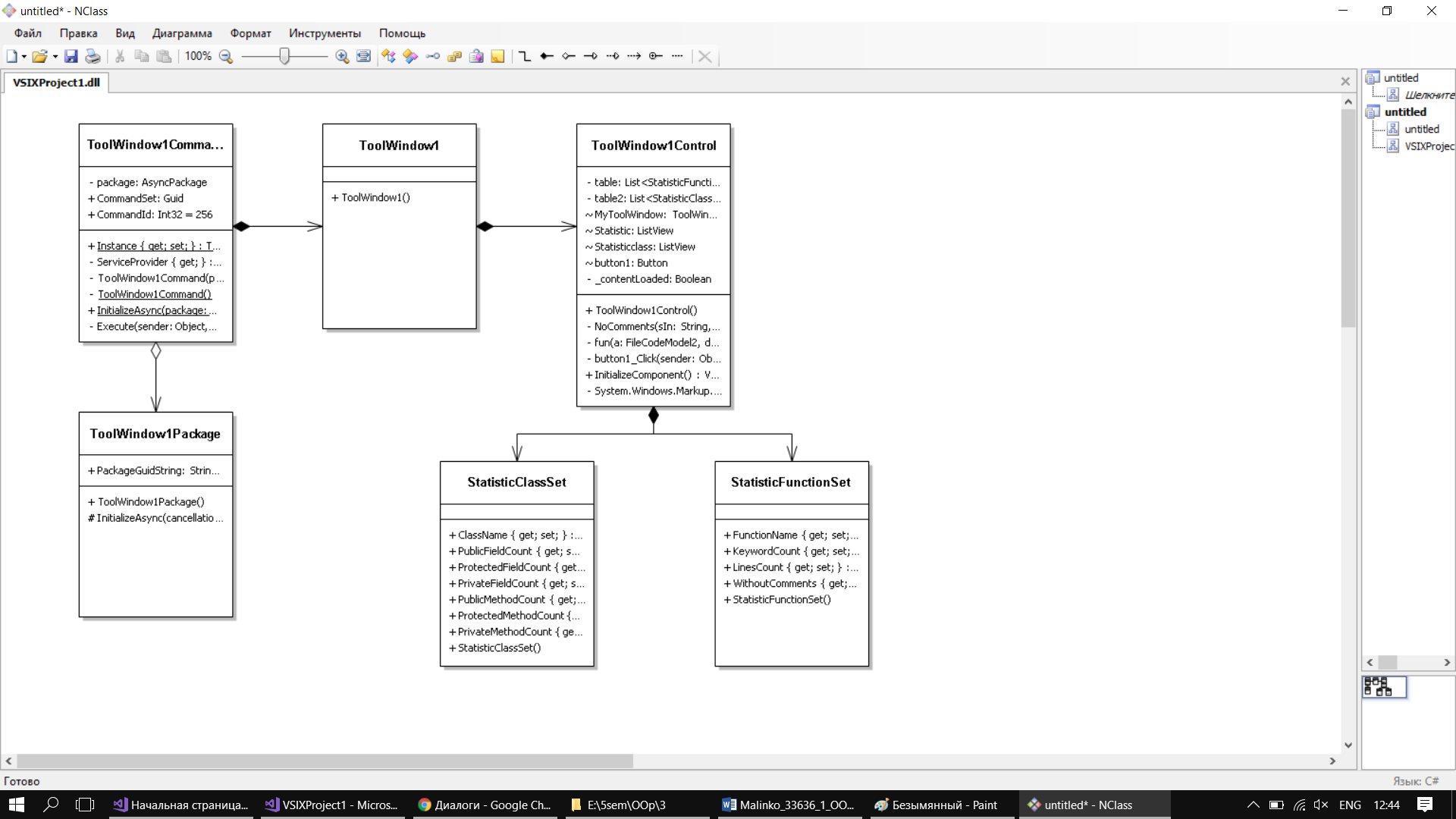
The second table contains all the information about the classes declared in the code - the number of different methods and fields.

**Work results**









**Source**

namespace VSIXProject1

{

using EnvDTE;

using EnvDTE80;

using Microsoft.VisualStudio.Shell;

using Microsoft.VisualStudio.Shell.Interop;

using System;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

using System.Windows;

using System.Windows.Controls;

using System.Text.RegularExpressions;

/// <summary>

/// Interaction logic for ToolWindow1Control.

/// </summary>

public class StatisticFunctionSet

{

public string FunctionName { get; set; }

public string KeywordCount { get; set; }

public string LinesCount { get; set; }

public string WithoutComments { get; set; }

}

public class StatisticClassSet

{

public string ClassName { get; set; }

public string PublicFieldCount { get; set; }

public string ProtectedFieldCount { get; set; }

public string PrivateFieldCount { get; set; }

public string PublicMethodCount { get; set; }

public string ProtectedMethodCount { get; set; }

public string PrivateMethodCount { get; set; }

}

public partial class ToolWindow1Control : UserControl

{

/// <summary>

/// Initializes a new instance of the <see cref="ToolWindow1Control"/> class.

/// </summary>

private List<StatisticFunctionSet> table;

private List<StatisticClassSet> table2;

public ToolWindow1Control()

{

this.InitializeComponent();

table = new List<StatisticFunctionSet>();

table2 = new List<StatisticClassSet>();

Statistic.ItemsSource = table;

Statisticclass.ItemsSource = table2;

}

/// <summary>

/// Handles click on the button by displaying a message box.

/// </summary>

/// <param name="sender">The event sender.</param>

/// <param name="e">The event args.</param>

[SuppressMessage("Microsoft.Globalization", "CA1300:SpecifyMessageBoxOptions", Justification = "Sample code")]

[SuppressMessage("StyleCop.CSharp.NamingRules", "SA1300:ElementMustBeginWithUpperCaseLetter", Justification = "Default event handler naming pattern")]

private string NoComments(string sIn, int lang)

{

//var fullClear = @"((("")(([^\n;])|((?<=\\)[^$])){0,})|((\@"")(?(?="")""{2}|[^""]){0,}[""])|(\/\\*([^\*]|(\\*+([^\/]))){0,}((\\*\/)|($)))|((\/)(\/)([^\n]\*(\\)(\n))\*[^\n]\*))";

//var stringCS = @"((("")(([^\n;])|((?<=\\)[^$])){0,})|((\@"")(?(?="")""{2}|[^""]){0,}[""]))";

var stringCPP = @"""((\\"")|(\\\n)|[^""\n])\*(""|[^\n](?=\n))";

//var stringCPP = @" ""((\\"")|(\\\n)|[^""\n])\*(""|\n) ";

var stringCS = @"""((\\"")|(\\\n)|[^""\n])\*(""|\n)|((\@"")(?(?="")""{2}|[^""]){0,}[""]))";

//var stringCS = @" ""((\\"")|(\\\n)|[^""\n])\*(""|\n) | ((\@"")(?(?="")""{2}|[^""]){0,}[""])) ";

var commentsCPP = @"((\/\\*([^\*]|(\\*+([^\/]))){0,}((\\*\/)|($)))|((\/)(\/)([^\n]\*(\\)(\n))\*[^\n]\*))";

//var commentsCPP = @"( (\/\\*([^\*]|(\\*+([^\/]))){0,}((\\*\/) | ($)))|((\/)(\/)([^\n]\*(\\)(\n))\*[^\n]\*) )";

var commentsCS = @"(((\/)(\/)(\/)[^\n]\*)|(\/\\*([^\*]|(\\*+([^\/]))){0,}((\\*\/)|($)))|((\/)(\/)([^\n]\*(\\)(\n))\*[^\n]\*))";

//var commentsCS = @"( ((\/)(\/)(\/)[^\n]\*) | (\/\\*([^\*]|(\\*+([^\/]))){0,}((\\*\/)|($))) | ((\/)(\/)([^\n]\*(\\)(\n))\*[^\n]\*) )";

switch (lang)

{

case (1):

{

var fullClear = stringCPP + "|" + commentsCPP;

Regex rgx = new Regex(fullClear);

string noComments = rgx.Replace(sIn, "");

return noComments;

}

case (2):

{

var fullClear = stringCS + "|" + commentsCS;

Regex rgx = new Regex(fullClear);

string noComments = rgx.Replace(sIn, "");

return noComments;

}

default:

{

return "";

}

}

}

private void fun(FileCodeModel2 a, Document document)

{

Dispatcher.VerifyAccess();

foreach (CodeElement element in a.CodeElements)

{

if (element.Kind == vsCMElement.vsCMElementFunction)

{

CodeFunction function = element as CodeFunction;

TextPoint start = function.GetStartPoint(vsCMPart.vsCMPartHeader);

TextPoint finish = function.GetEndPoint();

string fullSource = start.CreateEditPoint().GetText(finish);

int openCurlyBracePos = fullSource.IndexOf('{');

if (openCurlyBracePos > -1)

{

string pattern;

string clearcode;

var nor = @"(\r)";// удаляем \r, считаем все строки

Regex rgx = new Regex(nor);

fullSource = rgx.Replace(fullSource, "");

string enter = @"(\n\t\*)";

int line = Regex.Matches(fullSource, enter).Count;

line++;

string lines = line.ToString();

switch (document.Language)

{

case "C/C++":

pattern = @"\b(auto|double|int|struct|break|else|long|switch|case|enum|register|typedef|char|extern|return|union|const|float|short|unsigned|continue|for|signed|void|default|goto|sizeof|volatile|do|if|static|while)\b";

clearcode = NoComments(fullSource, 1);// c/c++

break;

case "CSharp":

pattern = @"\b(abstract|as|base|bool|break|byte|case|catch|char|checked|class|const|continue|decimal|default|delegate|do|double|else|enum|event|explicit|extern|false|finally|fixed|float|for|foreach|goto|if|implicit|in|int|interface|internal|is|lock|long|namespace|new|null|object|operator|out|override|params|private|protected|public|readonly|ref|return|sbyte|sealed|short|sizeof|stackalloc|static|string|struct|switch|this|throw|true|try|typeof|uint|ulong|unchecked|unsafe|ushort|using|using static|virtual|void|volatile|while)\b";

clearcode = NoComments(fullSource, 2);// c#

break;

default:

table.Add(new StatisticFunctionSet()

{

FunctionName = "ERROR",

KeywordCount = "ERROR",

LinesCount = "ERROR",

WithoutComments = "ERROR"

}

);

return;

}

var noVoid = @"(\n\t\*)(\n\t\*)+";

Regex rgx2 = new Regex(noVoid);

clearcode = rgx2.Replace(clearcode, "\n\t");

string prototype = clearcode.Substring(0, openCurlyBracePos).Trim();

int output = Regex.Matches(clearcode, pattern).Count;

string outputs = output.ToString();

string noVoidstring = rgx2.Replace(clearcode, "\n\t");

int linesclear = Regex.Matches(noVoidstring, enter).Count;

linesclear++;

string linesclears = linesclear.ToString();

table.Add(new StatisticFunctionSet()

{

FunctionName = prototype,

KeywordCount = outputs,

LinesCount = lines,

WithoutComments = linesclears

}

);

}

}

if (element.Kind == vsCMElement.vsCMElementClass)

{

int meth\_publ, meth\_prot, meth\_priv, field\_publ, field\_prot, field\_priv;

meth\_publ = meth\_prot = meth\_priv = field\_publ = field\_prot = field\_priv = 0;

CodeClass classElement = element as CodeClass;

foreach (CodeElement elem in classElement.Children)

{

if (elem.Kind == vsCMElement.vsCMElementFunction)

{

CodeFunction funelem = elem as CodeFunction;

if (funelem.Access == vsCMAccess.vsCMAccessPublic)

{

meth\_publ++;

}

if (funelem.Access == vsCMAccess.vsCMAccessProtected)

{

meth\_prot++;

}

if (funelem.Access == vsCMAccess.vsCMAccessPrivate)

{

meth\_priv++;

}

}

if (elem.Kind == vsCMElement.vsCMElementVariable)

{

CodeVariable varelem = elem as CodeVariable;

if (varelem.Access == vsCMAccess.vsCMAccessPublic)

{

field\_publ++;

}

if (varelem.Access == vsCMAccess.vsCMAccessProtected)

{

field\_prot++;

}

if (varelem.Access == vsCMAccess.vsCMAccessPrivate)

{

field\_priv++;

}

}

}

table2.Add(new StatisticClassSet()

{

ClassName = element.FullName,

PublicFieldCount = field\_publ.ToString(),

ProtectedFieldCount = field\_prot.ToString(),

PrivateFieldCount = field\_priv.ToString(),

PublicMethodCount = meth\_publ.ToString(),

ProtectedMethodCount = meth\_prot.ToString(),

PrivateMethodCount = meth\_priv.ToString()

}

);

}

}

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

table.Clear();

table2.Clear();

DTE2 dte; try

{

dte = (DTE2)ServiceProvider.GlobalProvider.GetService(typeof(DTE));

ProjectItem item = dte.ActiveDocument.ProjectItem;

Document document = dte.ActiveDocument;

if (item.IsOpen)

{

FileCodeModel2 model = (FileCodeModel2)item.FileCodeModel;

if (model != null)

fun(model, document);

}

}

catch (Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.ToString());

}

/\*VsShellUtilities.ShowMessageBox(ServiceProvider.GlobalProvider, "Let's check Output Windows!",

"Testing CodeElements Loop", OLEMSGICON.OLEMSGICON\_INFO,

OLEMSGBUTTON.OLEMSGBUTTON\_OK,

OLEMSGDEFBUTTON.OLEMSGDEFBUTTON\_FIRST);\*/

Statistic.Items.Refresh();

Statisticclass.Items.Refresh();

}

}

}