МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,

СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА» (СПбГУТ)

Санкт-Петербургский колледж телекоммуникаций им. Кренкеля

##### Лабораторно-практическая работа № 21

по профессиональному модулю

**ПМ.01 «ПРИКЛАДНОЕ ПРОГРАММИРОВАНИЕ»**

Студент гр. 502\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Савва В.И.

(подпись)

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Параничев

(оценка и подпись)

Санкт-Петербург

2022 год

**Разработка приложения для поиска данных на основе LINQ**

Код в Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

using WinForm\_CSharp\_\_Converted\_Data\_System;

namespace laba21

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form\_Main());

}

}

}

Код в Form\_Main.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using Excel = Microsoft.Office.Interop.Excel;

using System.IO;

namespace WinForm\_CSharp\_\_Converted\_Data\_System

{

public partial class Form\_Main : Form

{

/// <summary>

/// form's constants (постоянные класса формы)

/// </summary>

public const short MAX\_NUM\_OF\_DICTIONARY\_VALUES = 1000;

/// <summary>

/// application, book, list for active Excel's data

/// (хэндл приложения, книги, листа данных пользователя Excel)

/// </summary>

private Excel.Application \_xlsapp\_App;

private Excel.Workbook \_xlsbook\_Book;

/// <summary>

/// dictionary data to be extracted from Excel

/// (данные словаря, извлекаемые из таблиц Excel)

/// </summary>

private string[] \_str\_data\_values;

private short \_sh\_data\_count;

/// <summary>

/// input dictionary view

/// (представление словаря входных данных)

/// </summary>

private short[] \_sh\_input\_keys;

private string[] \_str\_input\_values;

private short \_sh\_input\_count;

/// <summary>

/// index of input tasks (индекс задания входных данных)

/// </summary>

private short \_sh\_input\_task\_index;

/// <summary>

/// form constructor (конструктор формы)

/// </summary>

public Form\_Main()

{

InitializeComponent();

// empty data before filing (данных нет, пока они не будут заполнены)

\_sh\_data\_count = 0;

\_sh\_input\_count = 0;

// allocation for addmissible range

// (динамическое выделение памяти для разрешенного диапазона значений)

\_str\_data\_values = new string[MAX\_NUM\_OF\_DICTIONARY\_VALUES];

\_sh\_input\_keys = new short[MAX\_NUM\_OF\_DICTIONARY\_VALUES];

\_str\_input\_values = new string[MAX\_NUM\_OF\_DICTIONARY\_VALUES];

}

/// <summary>

/// Form loading event (Событите при загрузке формы)

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Form\_Main\_Load(object sender, EventArgs e)

{

this.\_sh\_input\_task\_index = 0;

this.ComboBox\_Input.SelectedIndex = this.\_sh\_input\_task\_index;

this.Open\_Excel();

this.Read\_Data(this.\_sh\_input\_task\_index);

}

/// <summary>

/// Stop & close Excel‐server at form closing

/// (необходимо остановить и закрыть Excel‐сервер при закрытии формы)

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Form\_Main\_FormClosing(object sender,

FormClosingEventArgs e)

{

this.Close\_Excel();

}

/// <summary>

/// Read data using index task for input list

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_Read\_Click(object sender, EventArgs e)

{

this.Read\_Data(this.\_sh\_input\_task\_index);

this.ListBox\_Input.Items.Clear();

for (short sh\_i = 0; sh\_i < this.\_sh\_input\_count; sh\_i++)

{

this.ListBox\_Input.Items.Add(this.\_str\_input\_values[sh\_i]);

}

}

/// <summary>

/// Run query filled in evaluate list if its possible

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_RunQuery\_Click(object sender, EventArgs e)

{

string[] str\_values = new string[this.\_sh\_input\_count];

for (short sh\_i = 0; sh\_i < this.\_sh\_input\_count; sh\_i++)

{

str\_values[sh\_i] = this.\_str\_input\_values[sh\_i];

}

var v\_sampling =

from strs in str\_values

where (IsCorrespond(strs, (string)"ы", 7, 0) &&

IsCorrespond(strs, (string)"а", 0, 15)

) ||

IsCorrespond(strs, (string)"С", 6, 8)

orderby strs ascending

select strs;

this.ListBox\_Output.Items.Clear();

this.ListBox\_Output.Items.Add((string)"‐‐‐ v\_sampling ‐‐‐");

foreach (var v\_i in v\_sampling)

{

this.ListBox\_Output.Items.Add(v\_i);

}

this.ListBox\_Output.Items.Add((string)"‐‐‐ v\_subsampling ‐‐‐");

// foreach(var v\_i in v\_subsampling)

// {

// this.ListBox\_Output.Items.Add(v\_i);

// }

}

}

}

Код в программ Program\_Solution.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using Excel = Microsoft.Office.Interop.Excel;

using System.IO;

namespace WinForm\_CSharp\_\_Converted\_Data\_System

{

public partial class Form\_Main

{

/// <summary>

/// Open Excel‐server

/// </summary>

public short Open\_Excel()

{

// только если приложение xls еще не запущено для данного приложения

if (\_xlsapp\_App == null)

{

\_xlsapp\_App = new Excel.Application();

string path = "C:\\Users\\u268\\Documents\\Савва.xls";

FileInfo fi = new FileInfo(path);

// если файл существует и найден

if (fi.Exists)

{

// инициализация сервера xls для приложения через COM

\_xlsbook\_Book = \_xlsapp\_App.Workbooks.Open(path);

Excel.Sheets xlssheets\_Sheets = \_xlsbook\_Book.Worksheets;

// считывание данных с 1‐го листа

Excel.Worksheet xlsworksheet\_FirstSheet =

xlssheets\_Sheets.get\_Item(1);

\_sh\_data\_count = 0;

for (short sh\_iter = 1; sh\_iter < MAX\_NUM\_OF\_DICTIONARY\_VALUES;

sh\_iter++)

{

string str\_cell = "B" + sh\_iter.ToString();

Excel.Range xlsrange\_Range =

xlsworksheet\_FirstSheet.get\_Range(str\_cell, str\_cell);

string str\_value = Convert.ToString(xlsrange\_Range.Value2);

// завершение цикла, если получено пустое значение

if (str\_value == null)

{

break;

}

// иначе наполнение данных в вектор

else

{

this.\_str\_data\_values[sh\_iter - 1] = str\_value;

this.\_sh\_data\_count++;

}

}

//MessageBox.Show(null, this.\_str\_data\_values[0], this.\_str\_data\_values[this.\_sh\_data\_count‐1], 0);

}

else

{

MessageBox.Show(null, "Указанный файл Excel не существует!", "Ошибка", 0);

}

}

// иначе ошибка из‐за попытки замещения

else

{

MessageBox.Show(null, "Новый сервер Excel не создан, т.к выполняется рабочий сервер !", "Ошибка", 0);

}

return 0;

}

/// <summary>

/// Close Excel‐server

/// </summary>

public void Close\_Excel()

{

if (\_xlsapp\_App != null)

{

\_xlsbook\_Book.Close(0);

\_xlsapp\_App.Quit();

\_xlsbook\_Book = null;

\_xlsapp\_App = null;

GC.Collect();

}

}

/// <summary>

/// Scan all data from Excel‐server

/// </summary>

public short Read\_Data(short sh\_index\_task)

{

if (sh\_index\_task < 0 || (sh\_index\_task != 0 && sh\_index\_task >=

this.ComboBox\_Input.Items.Count))

return -1;

this.ComboBox\_Input.SelectedIndex = sh\_index\_task;

switch (sh\_index\_task)

{

case 0:

for (short sh\_i = 0; sh\_i < this.\_sh\_data\_count; sh\_i++)

{

this.\_sh\_input\_keys[sh\_i] = sh\_i;

this.\_str\_input\_values[sh\_i] = this.\_str\_data\_values[sh\_i];

}

this.\_sh\_input\_count = this.\_sh\_data\_count;

break;

case 1:

break;

}

return 0;

}/// END OF METHOD READ\_DATA

//‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐

public bool IsCorrespond(string str\_query, string str\_find,

short sh\_min\_count, short sh\_max\_count)

{

if (

(sh\_min\_count == 0 || sh\_min\_count < str\_query.Count()) &&

(sh\_max\_count == 0 || sh\_max\_count > str\_query.Count()) &&

(str\_find == null || str\_query.Contains(str\_find))

)

{

return true;

}

else

{

return false;

}

}

//‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐

}

}

Результат:

