**Лабораторная работа №5 по курсу**

**«Базовые компоненты интернет-технологий»**

Выполнил: Саврасов П.А. Группа РТ5-31

**Описание задания лабораторной работы.**

Разработать программу, реализующую вычисление расстояния Левенштейна с использованием алгоритма Вагнера-Фишера.

**Текст программы на языке С#.**

Класс Form1:

**using** System;  
**using** System.Collections.Generic;  
**using** System.Linq;  
**using** System.Windows.Forms;  
**using** System.Diagnostics;  
  
**namespace** LR4  
{  
    **public** partial class Form1 : Form  
    {  
        **public** Form1(List<string> Words)  
        {  
              
            *Words2* = Words;  
            **InitializeComponent**();  
  
        }  
        List<string> *Words2* = **new** List<string>();  
          
        #**region** Empty Elements  
        void **ComboBox1SelectedIndexChanged**(object sender, EventArgs e)  
        {  
      
        }  
        void **ComboBox2SelectedIndexChanged**(object sender, EventArgs e)  
        {  
      
        }  
        void **TextBox1TextChanged**(object sender, EventArgs e)  
        {  
      
        }  
        void **ListBox2SelectedIndexChanged**(object sender, EventArgs e)  
        {  
      
        }  
        void **TextBox2TextChanged**(object sender, EventArgs e)  
        {  
      
        }  
  
        #**endregion**  
          
        #**region** Search  
        void **Button1Click**(object sender, EventArgs e)  
        {      
            **if**(*comboBox1*.Text=="") MessageBox.**Show**("Chose Quantity of treads.","Error!",**MessageBoxButtons**.*OK*,**MessageBoxIcon**.*Error*);  
            **else**  
            {  
                **if**(*comboBox2*.Text=="") MessageBox.**Show**("Chose Distance.","Error!",**MessageBoxButtons**.*OK*,**MessageBoxIcon**.*Error*);  
                **else**  
                {  
                    **if**(*textBox1*.Text=="") MessageBox.**Show**("Enter Word.","Error!",**MessageBoxButtons**.*OK*,**MessageBoxIcon**.*Error*);  
                    **else**  
                    {  
                        List<string> Rezult = **new** List<string>();  
                        **int** A = (**int**.**Parse**(*comboBox1*.Text));  
                        **int** B = (**int**.**Parse**(*comboBox2*.Text));  
                        string S = (*textBox1*.Text);  
                        Stopwatch St2 = **new** Stopwatch();  
                        St2.**Start**();  
                        ASearch ASH = **new** ASearch();  
                        Rezult = ASH.**ASearcher**(*Words2*, A, B, S);  
                        *textBox2*.Text=St2.Elapsed.**ToString**();  
                        *listBox2*.**BeginUpdate**();  
                        *listBox2*.Items.**Clear**();  
                        **foreach**(string STR **in** Rezult)  
                            *listBox2*.Items.**Add**(STR);  
                        *listBox2*.**EndUpdate**();  
                    }  
                }  
            }  
        }  
  
        #**endregion**  
    }  
}

Класс ASearch:

**using** System;  
**using** System.Collections.Generic;  
**using** System.Linq;  
  
**namespace** LR4  
{  
    class ASearch  
    {  
        List<string> *Rezult* = **new** List<string>();  
        **public** List<string> **ASearcher**(List<string> Words, **int** NumberOfTreads, **int** MaxDistance, string ComWord)  
        {  
            **int**[] TreadWords = **new** **int**[10];  
            List<string> Words2 = **new** List<string>();  
            List<string>[] TWord = **new** List<string>[10];                        // Обьявление массива списков для обработки  
            **for**(**int** i=0;i<NumberOfTreads;i++)  
            {  
                TWord[i]=**new** List<string>();  
            }  
            **char**[] Valid=**new** **char**[ComWord.Length];  
            Valid[0]=ComWord[0];  
            **for**(**int** i=0;i<ComWord.Length;i++)  
                **if**(Array.**IndexOf**(Valid,ComWord[i])<0)Valid[i]=ComWord[i];  
            **foreach**(string str **in** Words)                                        // Создание нового списка слов, подходящих для поиска по длине ип совпадению символов.  
            {  
                **bool** Length = **false**;  
                **bool** Match = **false**;  
                **if**(Math.**Abs**(str.Length-ComWord.Length)<MaxDistance+1) Match=**true**;  
                **char**[] ValidCh=**new** **char**[str.Length];  
                ValidCh[0]=str[0];  
                **int** DDist=0;  
                **for**(**int** i=0;i<str.Length;i++)  
                    **if**(Array.**IndexOf**(ValidCh,str[i])<0)ValidCh[i]=str[i];  
                **for**(**int** i=0; i<ComWord.Length;i++)  
                {  
                    **bool** NotMatch=**true**;  
                    **for**(**int** k=0; k<str.Length;k++)  
                    {  
                        **if**(Valid[i]==ValidCh[k]) NotMatch=**false**;                                  
                    }  
                    **if**(NotMatch==**true**)DDist=DDist+1;  
                }  
                **if**(DDist<=MaxDistance)Match = **true**;  
                **if**((Length==**true**)||(Match==**true**)) Words2.**Add**(str);  
                  
            }  
              
            **int** Ost = Words2.**Count**()%NumberOfTreads;                            // Деление списка на части  
            **for**(**int** i=0;i<NumberOfTreads;i++)  
            {  
                TreadWords[i]= Words2.**Count**()/NumberOfTreads;  
                **if**(Ost!=0) {TreadWords[i]=TreadWords[i]+1;Ost=Ost-1;}  
            }  
  
            **int** j = 1;  
            **foreach**(string str **in** Words2)  
            {  
                **if**(j<=TreadWords[0]) TWord[0].**Add**(str);  
                **else**  
                    **if**(j<=TreadWords[0]+TreadWords[1]) TWord[1].**Add**(str);  
                    **else**  
                        **if**(j<=TreadWords[0]+TreadWords[1]+TreadWords[2]) TWord[2].**Add**(str);  
                        **else**  
                            **if**(j<=TreadWords[0]+TreadWords[1]+TreadWords[2]+TreadWords[3]) TWord[3].**Add**(str);  
                            **else**  
                                **if**(j<=TreadWords[0]+TreadWords[1]+TreadWords[2]+TreadWords[3]+TreadWords[4]) TWord[4].**Add**(str);  
                                **else**  
                                    **if**(j<=TreadWords[0]+TreadWords[1]+TreadWords[2]+TreadWords[3]+TreadWords[4]+TreadWords[5]) TWord[5].**Add**(str);  
                                    **else**  
                                        **if**(j<=TreadWords[0]+TreadWords[1]+TreadWords[2]+TreadWords[3]+TreadWords[4]+TreadWords[5]+TreadWords[6]) TWord[6].**Add**(str);  
                                        **else**  
                                            **if**(j<=TreadWords[0]+TreadWords[1]+TreadWords[2]+TreadWords[3]+TreadWords[4]+TreadWords[5]+TreadWords[6]+TreadWords[7]) TWord[7].**Add**(str);  
                                            **else**  
                                                **if**(j<=TreadWords[0]+TreadWords[1]+TreadWords[2]+TreadWords[3]+TreadWords[4]+TreadWords[5]+TreadWords[6]+TreadWords[7]+TreadWords[8]) TWord[8].**Add**(str);  
                                                **else**  
                                                    **if**(j<=TreadWords[0]+TreadWords[1]+TreadWords[2]+TreadWords[3]+TreadWords[4]+TreadWords[5]+TreadWords[6]+TreadWords[7]+TreadWords[8]+TreadWords[9]) TWord[9].**Add**(str);  
                j=j+1;  
            }  
            Distance Dist = **new** Distance();  
            *Rezult* = Dist.**LDistance**(TWord,MaxDistance,ComWord,NumberOfTreads);  
            return *Rezult*;  
        }  
  
    }  
}

Класс Distance:

**using** System;  
**using** System.Collections.Generic;  
**using** System.Linq;  
**using** System.Threading;  
**namespace** LR4  
{      
    **public** class Distance  
    {  
        **public** List<string> *Rezult* = **new** List<string>();  
        **public** List<string> **LDistance**(List<string>[] TWord,**int** MaxD,string CWord, **int** NOfTr)  
        {  
            Data[] DataInput = **new** Data[NOfTr];                    // Создание массива классов-хранидища информации  
            Thread[] Sercher = **new** Thread[NOfTr];                // Запуск потоков параллельного поиска + заполнение хранилищ данных  
            **for**(**int** i=0; i<NOfTr; i++)  
            {  
                DataInput[i] = **new** Data();  
                DataInput[i].*Dist* = MaxD;  
                DataInput[i].*Word* = CWord;  
                DataInput[i].*Words* = TWord[i];  
                DataInput[i].*RezultOfTr* = **new** List<string>();  
                DataInput[i].*Num* = i;  
                Sercher[i]=**new** Thread(**new** ParameterizedThreadStart(**TreadSearch**));  
                Sercher[i].**Start**(DataInput[i]);  
            }  
            **bool** IsCompleted=**false**;  
            **while**(IsCompleted==**false**)                            // Ожидание завершения работы всех потоков.  
            {  
                IsCompleted = **true**;  
                **for**(**int** i=1; i<NOfTr; i++)  
                    **if**(Sercher[i].IsAlive==**true**)IsCompleted=**false**;  
            }  
            **for**(**int** i=0; i<NOfTr; i++)  
            {  
                *Rezult*.**AddRange**(DataInput[i].*RezultOfTr*);  
            }  
            return *Rezult*;  
        }  
  
        **public** void **TreadSearch**(object DIn)  
        {  
            Data Dat1 = (Data)DIn;  
  
            string S = **null**;  
            **foreach**(string str **in** Dat1.*Words*)  
            {  
                **int** D  = **LDistance**(str,Dat1.*Word*);  
                **if** (D<=Dat1.*Dist*)  
                {  
                    S = S+"Tread № "+(Dat1.*Num*+1) + ": Distance between " + Dat1.*Word* + " and " + str + " equals " + D + ";\n";  
                    Dat1.*RezultOfTr*.**Add**(S);  
                }  
                S="";  
            }  
        }  
        **public** static **int** **LDistance**(string str1Param, string str2Param)  
        {   
            **if** ((str1Param == **null**) || (str2Param == **null**)) return -1;  
            **int** str1Len = str1Param.Length;  
            **int** str2Len = str2Param.Length;  
            **if** ((str1Len == 0) && (str2Len == 0)) return 0;  
            **if** (str1Len == 0) return str2Len;  
            **if** (str2Len == 0) return str1Len;  
            string str1 = str1Param.**ToUpper**();  
            string str2 = str2Param.**ToUpper**();  
            **int**[,] matrix = **new** **int**[str1Len + 1, str2Len + 1];  
            **for** (**int** i = 0; i <= str1Len; i++)  
                matrix[i, 0] = i;  
            **for** (**int** j = 0; j <= str2Len; j++)   
                matrix[0, j] = j;  
            **for** (**int** i = 1; i <= str1Len; i++)   
            {   
                **for** (**int** j = 1; j <= str2Len; j++)  
                {   
                    **int** symbEqual = ( (str1.**Substring**(i - 1, 1) == str2.**Substring**(j - 1, 1)) ? 0 : 1);  
                    **int** ins = matrix[i, j - 1] + 1;   
                    **int** del = matrix[i - 1, j] + 1;   
                    **int** subst = matrix[i - 1, j - 1] + symbEqual;  
                    matrix[i, j] = Math.**Min**(Math.**Min**(ins, del), subst);  
                    **if** ((i > 1) && (j > 1) && (str1.**Substring**(i - 1, 1) == str2.**Substring**(j - 2, 1)) && (str1.**Substring**(i - 2, 1) == str2.**Substring**(j - 1, 1)))  
                    {  
                        matrix[i, j] = Math.**Min**(matrix[i, j], matrix[i - 2, j - 2] + symbEqual);  
                    }  
                }   
            }  
            return matrix[str1Len, str2Len];   
        }  
    }  
}

Класс Data:

**using** System;  
**using** System.Collections.Generic;  
**using** System.Linq;  
  
**namespace** LR4  
{  
    class Data  
    {  
        **public** **int** *Num*;  
        **public** **int** *Dist*;  
        **public** List<string> *Words*;  
        **public** string *Word*;  
        **public**  List<string> *RezultOfTr*;  
    }  
}

**Результаты выполнения программы, экранные формы:**

Форма расширенного поиска:

