**Отчёт по лабораторной работе №6 по предмету «Базовые компоненты интернет-технологий»**

Подготовила: Гладова Анастасия

Группа: ИУ5-31

МГТУ имени Н.Э.Баумана

2017 г.

Разработать программу, реализующую работу с файлами.

1. Программа должна быть разработана в виде приложения Windows Forms на языке C#. По желанию вместо Windows Forms возможно использование WPF.
2. Добавить кнопку, реализующую функцию чтения файла в список слов List<string>.
3. Для выбора имени файла используется класс OpenFileDialog, который открывает диалоговое окно с выбором файла. Ограничить выбор только файлами с расширением «.txt».
4. Для чтения из файла рекомендуется использовать статический метод ReadAllText() класса File (пространство имен System.IO). Содержимое файла считывается методом ReadAllText() в виде одной строки, далее делится на слова с использованием метода Split() класса string. Слова сохраняются в список List<string>.
5. При сохранении слов в список List<string> дубликаты слов не записываются. Для проверки наличия слова в списке используется метод Contains().
6. Вычислить время загрузки и сохранения в список с использованием класса Stopwatch (пространство имен System.Diagnostics). Вычисленное время вывести на форму в поле ввода (TextBox) или надпись (Label).
7. Добавить на форму поле ввода для поиска слова и кнопку поиска. При нажатии на кнопку поиска осуществлять поиск введенного слова в списке. Слово считается найденным, если оно входит в элемент списка как подстрока (метод Contains() класса string).
8. Добавить на форму список (ListBox). Найденные слова выводить в список с использованием метода «название\_списка.Items.Add()». Вызовы метода «название\_списка.Items.Add()» должны находится между вызовами методов «название\_списка.BeginUpdate()» и «название\_списка. EndUpdate()».
9. Вычислить время поиска с использованием класса Stopwatch. Вычисленное время вывести на форму в поле ввода (TextBox) или надпись (Label).

**Диаграмма классов:**

**Текст программы:**

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Reflection;

namespace bkit\_lab\_6

{

class Program

{

//объявление делегата - площадь или периметр

delegate void AreaOrPerimeter(int a, int b, string units);

//площадь

static void Area(int width, int height, string units)

{

Console.WriteLine("Площадь прямоугольника: " + width \* height + " (" + units + ")^2");

}

//найти площадь или периметр

static void FindAreaOrPerimeter(int a, int b, string un,

AreaOrPerimeter p)

{

p(a, b, un);

}

//Action - обобщенный делегат, его определение Action<int, int, int>

//public delegate void Action<in T1, in T2, in T3>(T1 arg1, T2 arg2, T3 arg3);

//В отличие от делегата Func, у которого последний обобщенный параметр задает тип выходного значения,

//все обобщенные параметры делегата Action задают типы входных параметров.Обобщенный делегат Action возвращает значение типа void.

//найти площадь или периметр Action

static void FindAreaOrPerimeterAction(int a, int b, string un,

Action<int, int, string> f)

{

f(a, b, un);

}

//получить атрибут свойства

public static bool GetPropertyAttribute(PropertyInfo checkType, Type

attributeType, out object attribute)

{

bool Result = false;

attribute = null;

var isAttribute = checkType.GetCustomAttributes(attributeType,

false);

if (isAttribute.Length > 0)

{

Result = true;

attribute = isAttribute[0];

}

return Result;

}

static void Main(string[] args)

{

Console.WriteLine(" Использование делегатов");

//найти площадь или периметр

//значения

Console.WriteLine("Параметры прямоугольника: 3м, 4м");

//площадь

FindAreaOrPerimeter(3, 4, "м", Area);

//периметр

FindAreaOrPerimeter(3, 4, "м", (int a, int b, string un) =>

{

Console.WriteLine("Периметр прямоугольника: " + ((a + b) \* 2) + ' ' + un);

});

//найти площадь или периметр Action

Console.WriteLine("\n Использование делегата Action");

//значения

Console.WriteLine("Параметры прямоугольника: 3м, 8м");

//площадь

FindAreaOrPerimeterAction(3, 8, "м", Area);

//периметр

FindAreaOrPerimeterAction(3, 8, "м", (int a, int b, string un) =>

{

Console.WriteLine("Периметр прямоугольника: " + ((a + b) \* 2) + ' ' + un);

});

Type objType = typeof(Box);

//Информация о классе

Console.WriteLine("\n Информация о классе:");

Console.WriteLine(" Конструкторы:");

foreach (var constr in objType.GetConstructors())

Console.WriteLine(constr);

Console.WriteLine(" Методы:");

foreach (var meth in objType.GetMethods())

Console.WriteLine(meth);

Console.WriteLine(" Свойства:");

foreach (var prop in objType.GetProperties())

Console.WriteLine(prop);

Console.WriteLine(" Свойства, помеченные атрибутом:");

foreach (var prop in objType.GetProperties())

{

object attrObj;

if (GetPropertyAttribute(prop, typeof(AttributeClass), out

attrObj))

{

AttributeClass attr = attrObj as AttributeClass;

Console.WriteLine(prop.Name + " - " + attr.Description);

}

}

//вызов метода с использованием рефлексии

Console.WriteLine("\n Запуск метода");

Console.WriteLine("Напишите имя метода (внимание на регистр!):");

string methodName = Console.ReadLine();

Box box1 = new Box(1, 2, 3);

Type t = box1.GetType();

object[] parameters = new object[] { };

Console.WriteLine(t.InvokeMember(methodName, BindingFlags.InvokeMethod, null, box1, parameters));

Console.ReadLine();

}

}

}

**Box.cs**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Reflection;

namespace bkit\_lab\_6

{

//класс коробка ящик a\*b\*c

class Box

{

int width; //ширина

int height; //высота

int lenght; //длина

public Box(int a, int b, int c)

{

width = a;

height = b;

lenght = c;

}

//высота height

[AttributeClass("Details for height")]

public int propertyheight

{

get { return height; }

set { this.height = value; }

}

//ширина width

[AttributeClass("Details for width")]

public int propertywidth

{

get { return width; }

set { this.width = value; }

}

//свойства длины

public int propertylenght

{

get { return lenght; }

set { this.lenght = value; }

}

//вычисление объема V=a\*b\*c

public int Volume()

{

return this.width \* this.height \* this.lenght;

}

//вычисление площади поверхности S=2\*ab\*bc\*ac

public int AreaOfSurface()

{

return 2 \* (this.width \* this.height + this.lenght \* this.height + this.width \* this.lenght);

}

}

}

**AttributeClass.cs**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

namespace bkit\_lab\_6

{

[AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited = false)]

//класс атрибут

public class AttributeClass : Attribute

{

public AttributeClass() { }

public AttributeClass(string str)

{

Description = str;

}

public string Description //описание

{

set; //задать

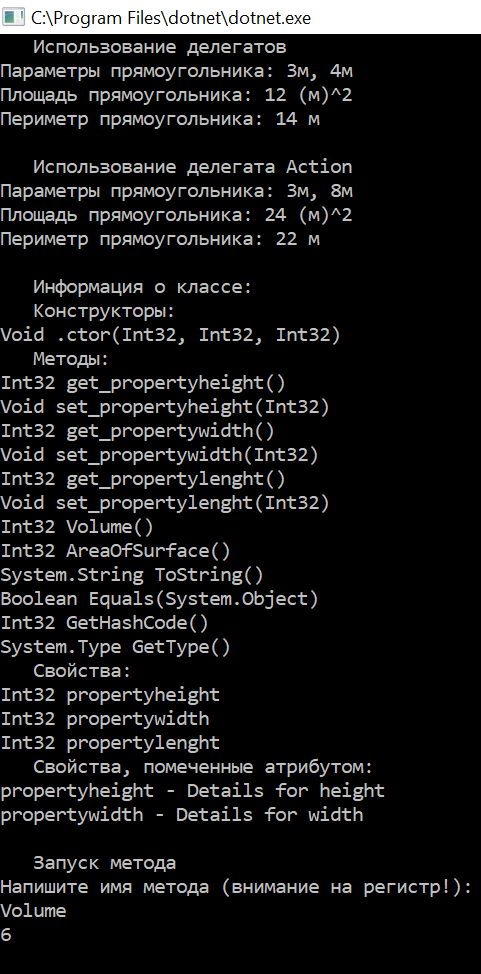
get; //получить

}

}

}

**Экранные формы с примерами выполнения программы:**

****