

Подписано электронной подписью:
Вержицкий Данил Григорьевич
Должность: Директор КГПИ ФГБОУ ВО «КемГУ»
Дата и время: 2024-02-21 00:00:00
471086fad29a3b30e244c728abc3661ab35c9d50210dcf0e75e03a5b6fdf6436
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Кемеровский государственный университет»
Новокузнецкий институт (филиал)

Факультет информатики, математики и экономики
Кафедра математики, физики и математического моделирования

А.Е. Паульзен, А.Д. Ульянов

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ВИЗУАЛЬНОГО ПРОГРАММИРОВАНИЯ

*Методические указания к выполнению лабораторных работ
для обучающихся по направлению подготовки
02.03.03 Математическое обеспечение и администрирование информационных систем,
профиль «Программное и математическое обеспечение информационных технологий»*

Новокузнецк
2020

УДК [378.147.88:004.43](072)
ББК 74.484(2Рос-4Кем)я73+32.973-018.1я73
П 21

Паульзен А.Е., А.Д. Ульянов

П 21 Инструментальные средства визуального программирования: методические указания к выполнению лабораторных работ для студентов факультета информатики, математики и экономики, обучающихся по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем, профиль «Программное и математическое обеспечение информационных технологий» / А.Е. Паульзен, А.Д. Ульянов; Новокузнецкий ин-т (фил.) Кемеров. гос. ун-та. – Новокузнецк : НФИ КемГУ, 2020 – 52 с.

Методические указания содержат краткие теоретические сведения, задания для выполнения на лабораторных работах по дисциплине «Инструментальные средства визуального программирования» и методические указания по их выполнению.

Методические указания предназначены для студентов всех форм обучения направлений 02.03.03 «Математическое обеспечение и администрирование информационных систем».

Рекомендовано на заседании
кафедры математики, физики и
математического моделирования
Протокол № 4 от 10.11.2020

Заведующий каф. МФММ



/ Е.В.Решетникова

Утверждено методической комиссией
факультета информатики, математики и
экономики
Протокол № 5 от 17.12.2020

Председатель методической комиссии
ФИМЭ



/Г.Н.Бойченко

УДК [378.147.88:004.43](072)
ББК 74.484(2Рос-4Кем)я73+32.973-018.1я73
П 21

© Паульзен Анна Евгеньевна
© Ульянов Артем Дмитриевич
© Федеральное государственное бюджетное
образовательное учреждение высшего
образования «Кемеровский государственный
университет»,
Новокузнецкий институт (филиал), 2020
текст представлен в авторской редакции

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
ЛАБОРАТОРНАЯ РАБОТА 1. ЛИНЕЙНЫЕ АЛГОРИТМЫ	5
ЛАБОРАТОРНАЯ РАБОТА 2. ЦИКЛИЧЕСКИЕ АЛГОРИТМЫ	11
ЛАБОРАТОРНАЯ РАБОТА 3. ОДНОМЕРНЫЕ МАССИВЫ	14
ЛАБОРАТОРНАЯ РАБОТА 4. ДВУМЕРНЫЕ МАССИВЫ	19
ЛАБОРАТОРНАЯ РАБОТА 5. МЕТОДЫ	22
ЛАБОРАТОРНАЯ РАБОТА 6. СТРУКТУРЫ И КОЛЛЕКЦИИ	27
ЛАБОРАТОРНАЯ РАБОТА 7. РАБОТА СО СТРОКАМИ	33
ЛАБОРАТОРНАЯ РАБОТА 8. ФАЙЛОВЫЙ ВВОД И ВЫВОД	36
ЛАБОРАТОРНАЯ РАБОТА 9. ФРАКТАЛЬНАЯ ГРАФИКА	42
ЛАБОРАТОРНАЯ РАБОТА 10. ПРОСТЕЙШАЯ АНИМАЦИЯ	45
ЛАБОРАТОРНАЯ РАБОТА 11. СОЗДАНИЕ ГРАФИЧЕСКОГО РЕДАКТОРА	47
РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА	52

ВВЕДЕНИЕ

В настоящих методических указаниях дано описание одиннадцати лабораторных работ, посвященных изучению языка C# и технологий разработки объектно-ориентированных программ в среде Visual Studio. В каждой лабораторной работе изложение теоретического материала сопровождается примерами, поясняющими суть вопроса, также разработан набор заданий для самостоятельного выполнения.

Основное содержание методических указаний по выполнению лабораторных работ ориентировано на обучение структурной методике программирования. Первые лабораторные работы предназначены для освоения программирования основных алгоритмических конструкций: линейной, ветвлению, циклам, а также способов составления подпрограмм. Последующие лабораторные работы посвящены работе с разными видами структур данных: массивы, коллекции, файлы. Завершающие лабораторные посвящены комплексному применению знаний для написания собственного текстового и графического редакторов.

ЛАБОРАТОРНАЯ РАБОТА 1. ЛИНЕЙНЫЕ АЛГОРИТМЫ

Задания к лабораторной работе 1

1. Реализовать линейный алгоритм для вычисления функции согласно своему варианту.
2. Значение каждой переменной задавать в текстовом поле.
3. Результат вычисления функции вывести в текстовом поле.
4. Оформить созданное приложение: на форме должны быть отражены ФИО и номер варианта, все компоненты должны иметь подпись.

Методические указания к лабораторной работе 1

Цель: научиться составлять каркас простейшего приложения в среде Visual Studio. Написать и отладить программу линейного алгоритма.

Ход работы

Запустить среду разработки **Visual Studio**.

После запуска появляется следующая стартовая страница, которая показана на рис. 1.

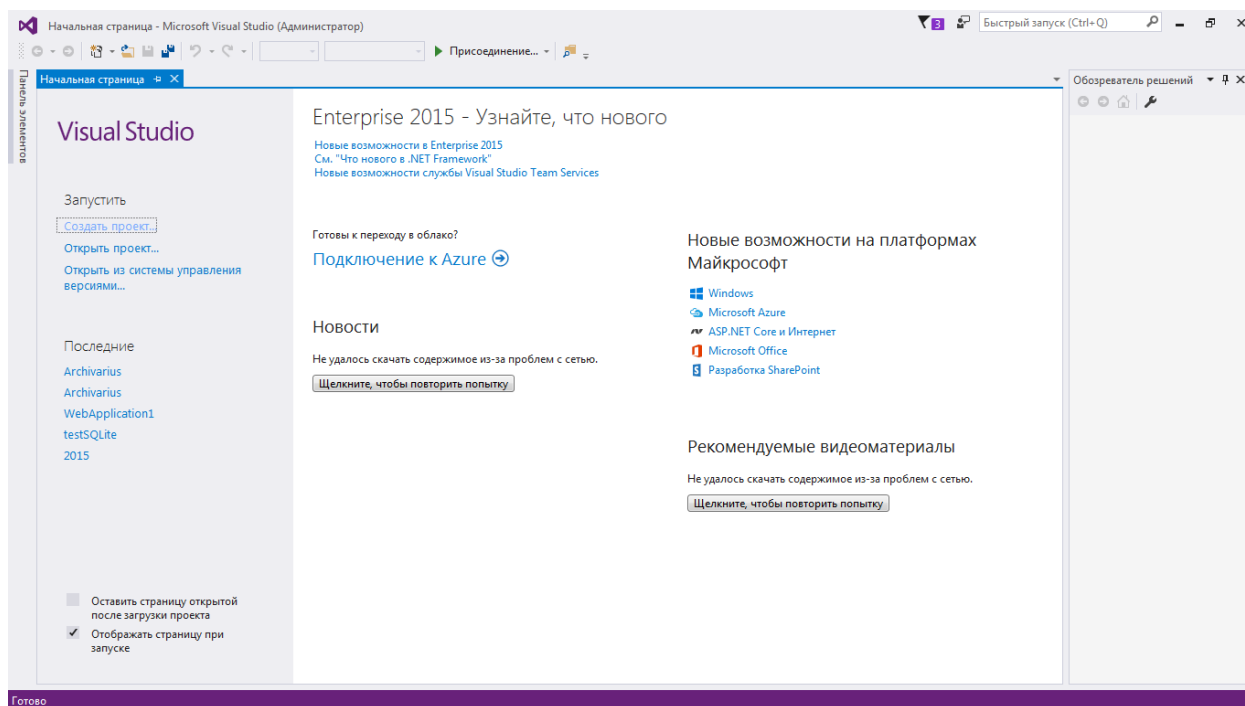


Рисунок 1 – Стартовая страница Visual Studio

Следующим шагом является создание нового проекта. Для этого в меню необходимо выбрать пункт меню **Создать проект...**, после чего появится окно (рис.2).

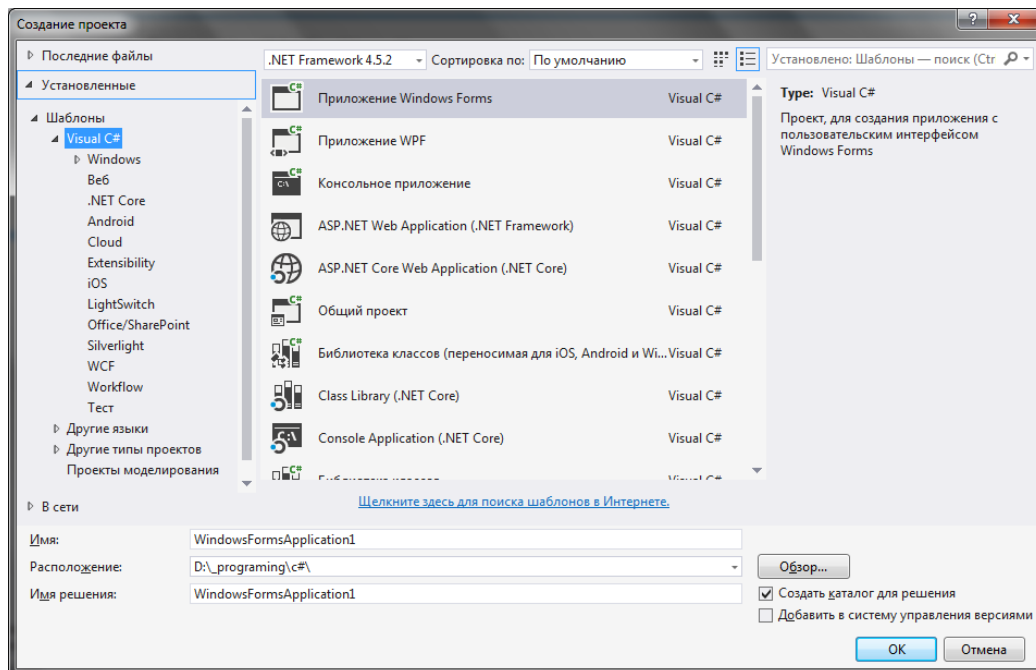


Рисунок 2 – Окно выбора типа создаваемого приложения

Среда Visual Studio отобразит окно «Создание проекта», в котором необходимо выбрать тип создаваемого проекта. В окне создания проекта следует развернуть узел Visual C# и на центральной панели выбрать Приложение Windows Form. Затем в поле редактора Имя следует ввести имя проекта. В поле Расположение указать путь размещения проекта. После нажатия кнопки ОК откроется окно, рис. 3.

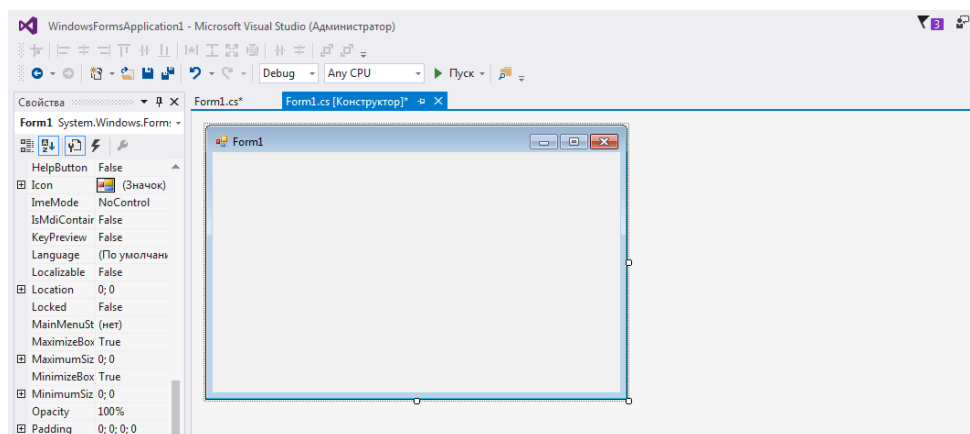


Рисунок 3 – Главное окно Visual Studio

Форма – будущее окно приложения, на котором будут размещаться элементы управления.

Окно свойств – объект, в котором приведены все основные свойства выделенного элемента управления. Если этого окна нет, его можно активировать в меню *Вид – Окно свойств*.

Сами элементы управления расположены на панели элементов, если этой панели нет, то ее можно показать, выбрав меню *Вид – Панель элементов*.

Обозреватель решений содержит список всех файлов, входящих в проект, включая добавленные изображения и служебные файлы. Активируется командой *Вид – Обозреватель решений*.

Переключение между формой и кодом можно с помощью команд: *Вид – Код* и *Вид – Конструктор*.

Элементы управления и их свойства

Изменение заголовка формы: свойство *Text*.

Элемент управления *Label* предназначен для размещения пояснительных надписей. Изменение надписи: свойство *Text*.

Элемент управления *TextBox* предназначен для ввода и вывода текстовой информации.

Контейнер *GroupBox* группирует элементы управления, свойство *Text* изменяет заголовок.

Обработчик событий нажатия кнопки

Дважды щелкнув мышью на кнопке, появится текст программы:

```
private void button1_Click(object sender, EventArgs e)
{
}
```

Эта функция и есть обработчик события нажатия кнопки. Код добавляется между фигурными скобками.

Ввод/вывод данных в программу

Для ввода или вывода текстовой информации часто используется элемент управления *TextBox*, через обращение к его свойству *Text*. Свойство *Text* хранит в себе строку введенных символов.

Чтение данных:

```
string s;  
s = textBox1.Text;  
int a;  
a = int.Parse(s);
```

Вывод данных:

```
int a;  
a = 10;  
textBox1.Text = a.ToString();
```

Стандартные математические функции

Math.Sin(a)– синус;

Math.Sinh(a)– гиперболический синус;

Math.Cos(a)– косинус (аргумент задается в радианах);

Math.Atan(a)– арктангенс (аргумент задается в радианах);

Math.Log(a)– натуральный логарифм;

Math.Exp(a)– экспонента;

Math.Pow(x,y)– возводит переменную x в степень y;

Math.Sqrt(a)– квадратный корень;

Math.Abs(a)– модуль числа;

Math.Truncate(a)– целая часть числа;

Math.Round(a)– округление числа.

В тригонометрических функциях все аргументы задаются в радианах.

Добавление символа перевода строки:

```
Environment.NewLine
```

Пример выполнения задания. Составить программу вычисления арифметического выражения при заданных X, Y, Z.

$$U = x^2 + 2y + z^4$$

Добавить на форму необходимые элементы управления. Пример приведен на рисунке 4.

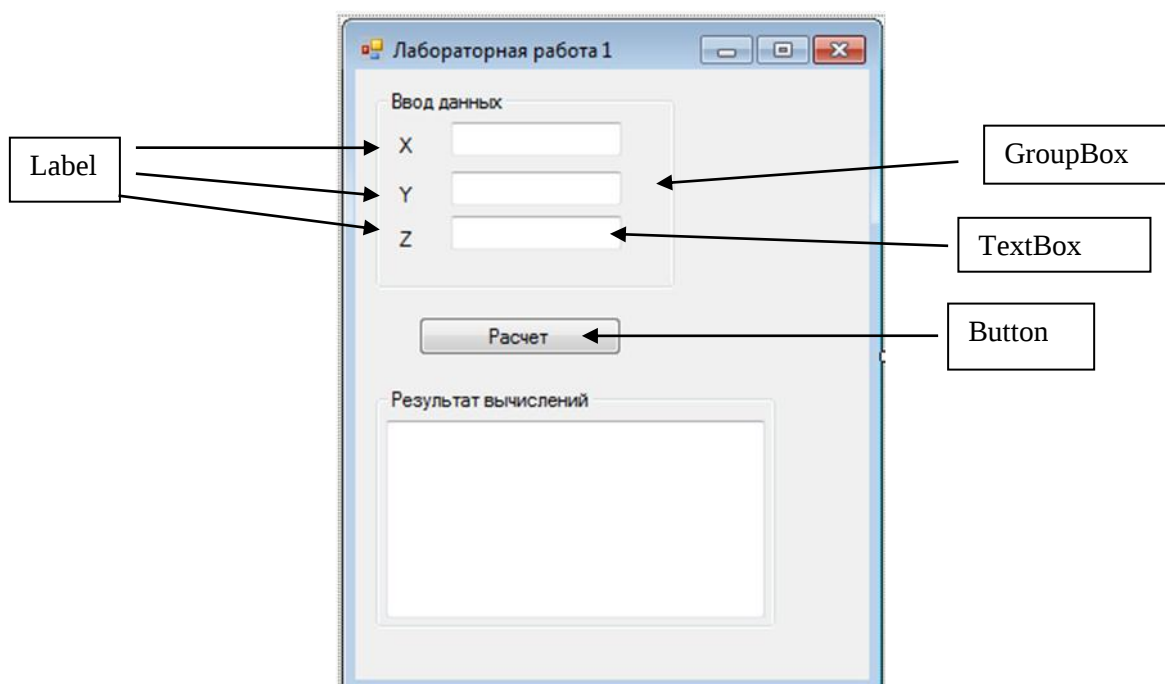


Рисунок 4 – Пример внешнего вида окна приложения

Формат вывода результата:

«При X = 2 Y = 4 Z = 3 U = 93»

Индивидуальные задания

В вариантах приведена функция для вычисления и контрольный пример: значения переменных и результат, который должен при этом получиться.

Вариант	Задание
1	$t = \frac{2 \cos\left(x - \frac{\pi}{6}\right)}{0.5 + \sin^2 y} \left(1 + \frac{z^2}{3 - z^2 / 5}\right).$ При $x = 14.26, y = -1.22, z = 3.5 \times 10^{-2}$ $t = 0.564849$.
2	$u = \frac{\sqrt[3]{8 + x - y ^2 + 1}}{x^2 + y^2 + 2} - e^{ x-y } (\operatorname{tg}^2 z + 1)^x.$ При $x = -4.5, y = 0.75 \times 10^{-4}, z = 0.845 \times 10^2$ $u = -55.6848$.
3	$v = \frac{1 + \sin^2(x + y)}{\left x - \frac{2y}{1 + x^2 y^2}\right } x^{ y } + \cos^2\left(\operatorname{arctg} \frac{1}{z}\right).$

4	$w = \cos x - \cos y ^{(1+2\sin^2 y)} \left(1 + z + \frac{z^2}{2} + \frac{z^3}{3} + \frac{z^4}{4} \right).$ При $x = 0.4 \times 10^4$, $y = -0.875$, $z = -0.475 \times 10^{-3}$ $w = 1.9873$.
5	$\alpha = \ln \left(y^{-\sqrt{ x }} \right) \left(x - \frac{y}{2} \right) + \sin^2 \operatorname{arctg}(z).$ При $x = -15.246$, $y = 4.642 \times 10^{-2}$, $z = 20.001 \times 10^2$ $\alpha = -182.036$.
6	$\beta = \sqrt{10 \left(\sqrt[3]{x} + x^{y+2} \right)} \left(\arcsin^2 z - x - y \right).$ При $x = 16.55 \times 10^{-3}$, $y = -2.75$, $z = 0.15$ $\beta = -38.902$.
7	$\gamma = 5 \operatorname{arctg}(x) - \frac{1}{4} \arccos(x) \frac{x + 3 x - y + x^2}{ x - y z + x^2}.$ При $x = 0.1722$, $y = 6.33$, $z = 3.25 \times 10^{-4}$ $\gamma = -172.025$.
8	$\varphi = \frac{e^{ x-y } x - y ^{x+y}}{\operatorname{arctg}(x) + \operatorname{arctg}(z)} + \sqrt[3]{x^6 + \ln^2 y}.$ При $x = -2.235 \times 10^{-2}$, $y = 2.23$, $z = 15.221$ $\varphi = 39.374$.
9	$\psi = \left x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}} \right + (y - x) \frac{\cos y - \frac{z}{(y - x)}}{1 + (y - x)^2}.$ При $x = 1.825 \times 10^2$, $y = 18.225$, $z = -3.298 \times 10^{-2}$ $\psi = 1.2131$.
10	$a = 2^{-x} \sqrt{x + \sqrt[4]{ y }} \sqrt[3]{e^{x-1/\sin z}}.$ При $x = 3.981 \times 10^{-2}$, $y = -1.625 \times 10^3$, $z = 0.512$ $a = 1.26185$.
11	$b = y^{\sqrt[3]{ x }} + \cos^3(y) \frac{ x - y \left(1 + \frac{\sin^2 z}{\sqrt{x + y}} \right)}{e^{ x-y } + \frac{x}{2}}.$ При $x = 6.251$, $y = 0.827$, $z = 25.001$ $b = 0.7121$.
12	$c = 2^{(y^x)} + (3^x)^y - \frac{y \left(\operatorname{arctg} z - \frac{\pi}{6} \right)}{ x + \frac{1}{y^2 + 1}}.$ При $x = 3.251$, $y = 0.325$, $z = 0.466 \times 10^{-4}$ $c = 4.025$.

13	$f = \frac{\sqrt[4]{y + \sqrt[3]{x-1}}}{ x-y (\sin^2 z + \operatorname{tg} z)}.$ При $x = 17.421, y = 10.365 \times 10^{-3}, z = 0.828 \times 10^5$ $f = 0.33056$.
14	$g = \frac{y^{x+1}}{\sqrt[3]{ y-2 } + 3} + \frac{x + \frac{y}{2}}{2 x+y } (x+1)^{-1/\sin z}.$ При $x = 12.3 \times 10^{-1}, y = 15.4, z = 0.252 \times 10^3$ $g = 82.8257$.
15	$h = \frac{x^{y+1} + e^{y-1}}{1 + x y - \operatorname{tg} z } \left(1 + y-x \right) + \frac{ y-x ^2}{2} - \frac{ y-x ^3}{3}.$ При $x = 2.444, y = 0.869 \times 10^{-2}, z = -0.13 \times 10^3$ $h = -0.49871$.

ЛАБОРАТОРНАЯ РАБОТА 2. ЦИКЛИЧЕСКИЕ АЛГОРИТМЫ

Задания к лабораторной работе 2

1. Создать приложение для вычисления значения функции и нахождения экстремумов функции:

$$y = \begin{cases} f1(x), & \text{если } x \leq 0 \\ f2(x), & \text{если } 0 < x \leq a \\ f3(x), & \text{если } x > a \end{cases}$$

с использованием оператора цикла на отрезке $[x_n; x_k]$ с шагом h .

Параметры x_n, x_k, h и a вводятся с клавиатуры.

2. В качестве функций $f1(x), f2(x), f3(x)$ выбрать произвольные алгебраические выражения. Для вывода формулы вычисляемого выражения использовать компонент PictureBox.

3. Оформить созданное приложение: на форме должны быть отражены ФИО, все компоненты должны иметь подпись.

Методические указания к лабораторной работе 2

Цель: изучение операторов цикла; создание циклических алгоритмов; написание алгоритмов на поиск экстремумов.

Цикл – разновидность управляющей конструкции высокоуровневых языков программирования, предназначенная для организации многократного исполнения набора инструкций.

Оператор while (цикл с предусловием)

while (условие)

```
{  
    действие;  
    действие2;  
    действие3;  
    ...  
}
```

Оператор do - while (цикл с постусловием)

```
do {  
    действие;  
    действие2;  
    действие3;  
    ...  
} while (условие);
```

Оператор for (цикл с параметром)

for (нач_знач; условие; счетчик)

```
{  
    действие;  
    действие2;  
    действие3;  
    ...  
}
```

Ход работы

Создать новый проект в среде Visual Studio.

Пример выполнения задания. Вычислить значения функции и найти экстремумы функции на отрезке $[x_n; x_k]$ с шагом x_h .

$$y = f(x) = \begin{cases} 2x + 2, & \text{если } x \leq 0 \\ \sqrt{x + 3}, & \text{если } 0 < x \leq a \\ \cos^2(x + 2), & \text{если } x > a \end{cases}.$$

Необходимые управляющие элементы приложения: поля для ввода начала и конца отрезка, шага вычисления функции, параметра a ; поля для вывода значения аргумента и функции, поля для вывода наибольшего и наименьшего значения функции.

Необходимо выполнять проверку введенных данных, в случае ошибки выводить диагностическое сообщение.

Пример пользовательского интерфейса приложения.

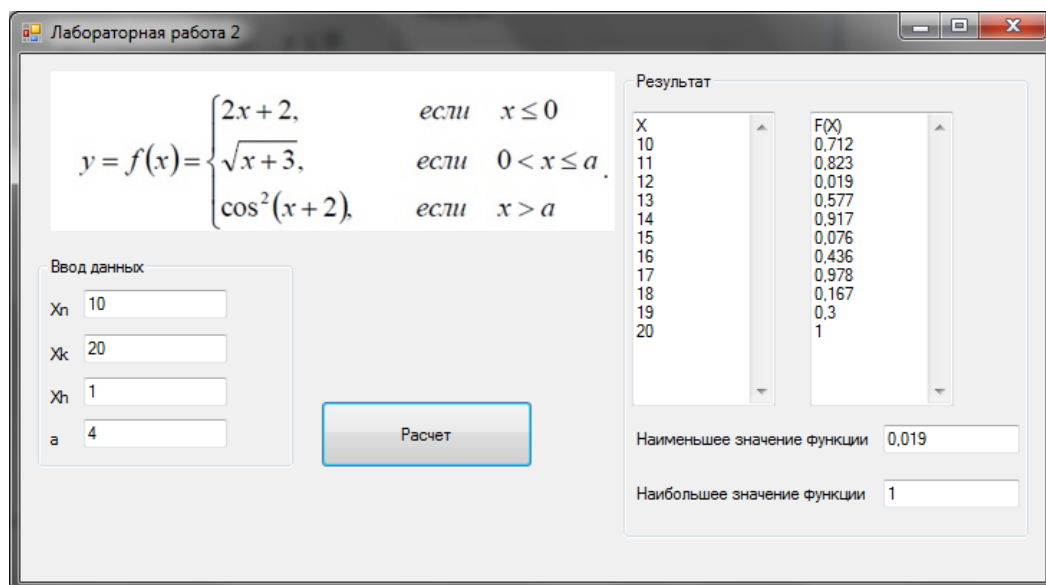


Рисунок 5 – Интерфейс приложения

ЛАБОРАТОРНАЯ РАБОТА 3. ОДНОМЕРНЫЕ МАССИВЫ

Задания к лабораторной работе 3

1. Сгенерировать и вывести массив случайных чисел.
2. Выполнить некоторое действие с этим массивом согласно своему варианту и после вывести результат, при этом исходный массив должен оставаться на форме.
3. Оформить созданное приложение: на форме должны быть отражены ФИО, все компоненты должны иметь подпись.

Методические указания к лабораторной работе 3

Цель: Изучить способы получения случайных чисел. Написать программу для работы с одномерными массивами.

Массив – набор элементов одного и того же типа, объединенных общим именем.

Рассмотрим в данной лабораторной работе одномерные массивы. *Одномерный массив* – это фиксированное количество элементов одного и того же типа, объединенных общим именем, где каждый элемент имеет свой номер. Нумерация элементов массива в C# начинается с нуля, то есть если массив состоит из 10 элементов, то его элементы будут иметь следующие номера: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Для объявления одномерного массива может использоваться одна из следующих форм записи:

тип[] имя_массива;

В этом случае описывается ссылка на одномерный массив, которая в дальнейшем может быть использована для адресации на уже существующий массив. Размер массива при таком объявлении не задается. Пример, в котором объявляется массив целых чисел с именем a:

```
int[] a;
```

Другая форма объявления массива включает и его инициализацию указанным количеством элементов:

```
тип[] имя_массива = new тип[размер];
```

В этом случае объявляется одномерный массив указанного типа и выделяется память под указанное количество элементов. Адрес данной области памяти записывается в ссылочную переменную. Элементы массива инициализируются значениями, которые по умолчанию приняты для данного типа: массивы числовых типов инициализируются нулями, строковые переменные – пустыми строками, символы – пробелами, объекты ссылочных типов – значением null. Пример такого объявления:

```
int[] a = new int[10];
```

Здесь выделяется память под 10 элементов типа int.

Наконец, третья форма записи дает возможность сразу инициализировать массив конкретными значениями:

```
тип[] имя_массива = {список инициализации};
```

При такой записи выделяется память под одномерный массив, размерность которого соответствует количеству элементов в списке инициализации. Адрес этой области памяти записан в ссылочную переменную. Значение элементов массива соответствует списку инициализации. Пример:

```
int[] a = { 0, 1, 2, 3 };
```

В данном случае будет создан массив a, состоящий из четырех элементов, и каждый элемент будет инициализирован очередным значением из списка.

Обращение к элементам массива происходит с помощью индекса: для этого нужно указать имя массива и в квадратных скобках – его номер. Например: a[0], b[10], c[i]. Следует помнить, что нумерация элементов начинается с нуля!

Случайные числа

Одним из способов инициализации массива является задание элементов через случайные числа. Для работы со случайными числами используют класс Random.

Необходимо создать объект для генерации случайных чисел.

```
Random rnd = new Random();
```

rnd – имя переменной типа Random.

Методы класса Random

Название	Описание
Next ()	Возвращает целое положительное число во всем положительном диапазоне типа int
Next (макс)	Возвращает целое положительное число в диапазоне [0, макс]
Next (мин, макс)	Возвращает целое положительное число в диапазоне [мин, макс]
NextDouble()	Возвращает вещественное положительное число в диапазоне [0, 1)

Получить очередное случайное число

```
int value = rnd.Next();
```

Получить случайное число (в диапазоне от 0 до 10)

```
int value = rnd.Next(0, 10);
```

Получить целое положительное число не больше максимального

```
int value = rnd.Next(10);
```

Дробные числа от 0 до 2 с точностью до сотых

```
Random rand = new Random();  
double temp;  
temp = Convert.ToDouble(rand.Next(200))/100;
```

Дробные числа от 0 до 10 с точностью до тысячных

```
Random rand = new Random();  
double temp;  
temp = Convert.ToDouble(rand.Next(10000))/1000
```

Массив случайных чисел

```
Random rand = new Random();  
int [] Mass = new int [20];  
for (int i = 0; i<20; i++)  
Mass[i] = rand.Next();
```

Индивидуальные задания

1. В массиве из 20 целых чисел найти наибольший элемент и поменять его местами с первым элементом.
2. В массиве из 10 целых чисел найти наименьший элемент и поменять его местами с предпоследним элементом.
3. Дан массив F , содержащий 18 элементов. Вычислить и вывести элементы нового массива по формуле $p_i = 0.13f^3 - 2.5f + 8$.
4. В массиве R , содержащем 25 элементов, заменить значения отрицательных элементов квадратами значений, значения положительных увеличить на 7, а нулевые значения оставить без изменения.
5. Дан массив A целых чисел, содержащий 30 элементов. Вычислить и вывести сумму тех элементов, которые кратны 5.
6. Дан массив A целых чисел, содержащий 30 элементов. Вычислить и вывести сумму тех элементов, которые нечетны и отрицательны.
7. Дан массив A целых чисел, содержащий 30 элементов. Вычислить и вывести количество и сумму тех элементов, которые делятся на 5 и не делятся на 7.
8. Дан массив A вещественных чисел, содержащий 25 элементов. Вычислить и вывести число отрицательных элементов и число членов, принадлежащих отрезку $[1,2]$.
9. Дан массив Z целых чисел, содержащий 35 элементов. Вычислить и вывести $R = S + P$, где S – сумма четных элементов, меньших 3, P – произведение нечетных элементов, больших 1.
10. Дан массив Q натуральных чисел, содержащий 20 элементов. Найти и вывести те элементы, которые при делении на 7 дают остаток 1, 2 или 5.
11. Дан массив, содержащий 10 элементов. Вычислить произведение элементов, стоящих после первого отрицательного элемента. Вывести исходный массив и результат вычислений.
12. Дан массив, содержащий 14 элементов. Вычислить сумму элементов, стоящих до первого отрицательного элемента. Вывести исходный

массив и результат вычислений.

13. Дан массив, содержащий 12 элементов. Все четные элементы сложить, вывести массив и результат.

14. Дан массив, содержащий 15 элементов. Все положительные элементы возвести в квадрат, а отрицательные умножить на 2. Вывести исходный и полученный массив.

15. Дан массив, содержащий 14 элементов. Все отрицательные элементы заменить на 3. Вывести исходный и полученный массив.

16. Массив задан датчиком случайных чисел на интервале $[-33, 66]$. Найти наименьший нечетный элемент.

17. Разработать программу, выводящую количество максимальных элементов в массиве из пятидесяти целочисленных элементов.

18. Разработать программу, циклически сдвигающую элементы целочисленного массива влево. Нулевой элемент массива ставится на последнее место, остальные элементы сдвигаются влево на одну позицию. Запрещается использовать второй массив.

19. Дан массив целых чисел из 30 элементов. Найдите все локальные максимумы. (Элемент является локальным максимумом, если он не имеет соседей, больших, чем он сам).

ЛАБОРАТОРНАЯ РАБОТА 4. ДВУМЕРНЫЕ МАССИВЫ

Задания к лабораторной работе 4

1. Выполнить задание согласно своему варианту.
2. Оформить созданное приложение: на форме должны быть отражены ФИО, все компоненты должны иметь подпись.

Методические указания к лабораторной работе 4

Цель: изучить свойства элемента управления DataGridView. Написать программу с использованием двумерных массивов.

Многомерные массивы имеют более одного измерения. Чаще всего используются двумерные массивы, которые представляют собой таблицы. Каждый элемент такого массива имеет два индекса, первый определяет номер строки, второй – номер столбца, на пересечении которых находится элемент. Нумерация строк и столбцов начинается с нуля.

Объявление массива

- тип[,] имя_массива = new тип[размер1, размер2];
- тип[,] имя_массива =
 {{элементы 1-ой строки}, ...,
 {элементы n-ой строки}};

Элемент управления DataGridView

При работе с двумерными массивами ввод и вывод информации на экран удобно организовывать в виде таблиц. Элемент управления DataGridView может быть использован для отображения информации в виде двумерной таблицы. Для обращения к ячейке в этом элементе необходимо указать номер строки и номер столбца. Например:

```
dataGridView1.Rows[2].Cells[7].Value = "*";
```

Этот код запишет во вторую строку и седьмой столбец знак звездочки.

Задание размеров таблицы

```
dataGridView1.ColumnCount = 5;  
dataGridView1.RowCount = 5;
```

Прочитать количество строк и столбцов

```
value = dataGridView1.ColumnCount;  
value = dataGridView1.RowCount
```

Присвоить значение ячейке

```
dataGridView1.Rows[0].Cells[0].Value = value;
```

где value – значение произвольного типа

Прочитать значение из ячейки

```
double value =  
Convert.ToDouble(dataGridView1.Rows[0].Cells[0].Value);
```

Заполнение DataGridView случайными числами

```
Random rand = new Random();  
int str = dataGridView1.RowCount;  
int stolb = dataGridView1.ColumnCount;  
for (int i = 0; i < str; i++)  
{  
    for (int j = 0; j < stolb; j++)  
        dataGridView1.Rows[i].Cells[j].Value = rand.Next(7, 100);  
}
```

Чтение в массив из DataGridView

```
int[,] Mass = new int[str, stolb];  
for (int i = 0; i < str; i++)  
{  
    for (int j = 0; j < stolb; j++)  
        Mass[i, j] =  
Convert.ToInt32(dataGridView1.Rows[i].Cells[j].Value);  
}
```

Индивидуальные задания

1. Дана матрица $A(3,4)$. Найти наименьший элемент в каждой строке матрицы. Вывести исходную матрицу и результаты вычислений.
2. Дана матрица $A(3,3)$. Вычислить сумму элементов второй строки и произведение элементов первого столбца. Вывести исходную матрицу и результаты вычислений.

3. Вычислить сумму элементов главной диагонали матрицы $B(10,10)$.
4. Дана матрица $F(15,15)$. Вывести среднее арифметическое элементов каждой строки.
5. Дана матрица $F(7,7)$. Найти наименьший элемент в каждом столбце. Вывести матрицу и найденные элементы.
6. Найти наибольший элемент главной диагонали матрицы $A(15,15)$ и вывести всю строку, в которой он находится.
7. Найти наибольшие элементы каждой строки матрицы $Z(16,16)$ и поместить их на главную диагональ. Вывести полученную матрицу.
8. Найти наибольший элемент матрицы $A(10,10)$ и записать нули в ту строку и столбец, где он находится. Вывести наибольший элемент, исходную и полученную матрицу.
9. Дана матрица $R(9,9)$. Найти наименьший элемент в каждой строке и записать его на место первого элемента строки. Вывести исходную и полученную матрицы.
10. Вычислить и вывести сумму элементов матрицы $A(12,12)$, расположенных над главной диагональю матрицы.
11. Найти номер столбца матрицы, в котором находится наименьшее количество положительных элементов.
12. Найти наименьший элемент главной диагонали матрицы $A(15,15)$ и вывести всю строку, в которой он находится.
13. Дана матрица $F(7,7)$. Найти наибольший элемент в каждом столбце. Вывести матрицу и найденные элементы.
14. Вычислить произведение элементов главной диагонали матрицы $B(10,10)$.
15. Дана матрица $A(5,4)$. Найти наибольший элемент в каждой строке матрицы. Вывести исходную матрицу и результаты вычислений.

ЛАБОРАТОРНАЯ РАБОТА 5. МЕТОДЫ

Задания к лабораторной работе 5

1. Выполнить задание согласно своему варианту.
2. Оформить созданное приложение: на форме должны быть отражены ФИО, все компоненты должны иметь подпись.

Методические указания к лабораторной работе 5

Цель лабораторной работы: научиться работать с методами, написать программу с использованием методов.

Метод – подпрограмма, выполняющая определенный набор операций.

[Тип возвращаемого значения] [Имя Метода] ([Параметры])

{

КОД_МЕТОДА

}

Тип определяет результат, который возвращает метод: это может быть любой тип, доступный в C#, а также ключевое слово void, если результат не требуется

Имя метода – это идентификатор, который будет использоваться для вызова метода. К идентификатору применяются те же требования, что и к именам переменных: он может состоять из букв, цифр и знака подчеркивания, но не может начинаться с цифры.

Параметры – это список переменных, которые можно передавать в метод при вызове. Каждый параметр состоит из типа и названия переменной. Параметры разделяются запятой.

Тело метода – это обычный программный код, за исключением того, что он не может содержать определения других методов и классов. Если метод должен возвращать какой-то результат, то обязательно в конце должно присутствовать ключевое слово return с возвращаемым значением. Если возвращение результатов не нужно, то использование ключевого слова return не обязательно, хотя и допускается.

Примеры методов

Функция, не возвращающая значения и не принимающая аргументы:

```
void printError()  
{  
    Console.WriteLine("Error! Press Key...");  
    Console.ReadKey();  
}
```

Функция, не возвращающая значения, но принимающая аргумент:

```
void printError(string s)  
{  
    Console.WriteLine("Error! " + s + "Press Key...");  
    Console.ReadKey();  
}
```

Функция, возвращающая значения и принимающая аргументы:

```
char shifr(char x, int shifr)  
{  
    x = (char)(x + shifr);  
    return x;  
}
```

Перегрузка методов

Язык C# позволяет создавать несколько методов с одинаковыми именами, но разными параметрами. Компилятор автоматически подберет наиболее подходящий метод при построении программы.

```
public void ui (string Name)  
{  
    Console.WriteLine("Имя пользователя: {0}",Name);  
}  
public void ui (string Name, string Family)  
{  
    Console.WriteLine("Имя пользователя: {0}\nФамилия  
пользователя: {1}",Name,Family);  
}  
public void ui (string Name, string Family, byte Age)  
{
```

```

        Console.WriteLine("Имя пользователя: {0}\nФамилия
пользователя: {1}\nВозраст: {2}", Name, Family, Age);
    }

```

Параметры по умолчанию

Язык C# начиная с версии 4.0 (Visual Studio 2010), позволяет задавать некоторым параметрам *значения по умолчанию* – так, чтобы при вызове метода можно было опускать часть параметров.

```

private void GetData(int Number, int Optional = 5)
{
    MessageBox.Show("Number: {0}", Number);
    MessageBox.Show("Optional: {0}", Optional);
}

```

В этом случае вызывать метод можно следующим образом:

```

GetData(10, 20);
GetData(10);

```

Параметры по умолчанию можно ставить только в правой части списка параметров, например, такая сигнатура метода компилятором принята не будет:

```

private void GetData(int Optional = 5, int Number)

```

Рекурсивные методы

Рекурсивным называют метод, который прямо (непосредственно) или косвенно вызывает самого себя.

Метод называют **косвенно рекурсивным**, если он содержит обращение к другому методу, содержащему прямой или косвенный вызов определяемого (первого) метода.

Если в теле метода явно используется обращение к этому методу, то имеет место прямая рекурсия. В этом случае говорят, что метод самовывзывающий (self-calling). Именно самовывзывающие методы будем называть **рекурсивными**, а для методов с косвенной рекурсией будем использовать термин **косвенно рекурсивные** методы.

Пример рекурсивного метода: вычисление факториала числа.

```

int Factorial(int x)
{

```

```

    if (x == 0)
    {
        return 1;
    }
    else
    {
        return x * Factorial(x - 1);
    }
}

```

Индивидуальные задания

1. Написать метод $\min(x, y)$, находящий минимальное значение из двух чисел. С его помощью найти минимальное значение из четырех чисел a, b, c, d.

2. Написать метод $\max(x, y)$, находящий максимальное значение из двух чисел. С его помощью найти максимальное значение из четырех чисел a, b, c, d.

3. Написать метод, вычисляющий значение n/x^n . С его помощью вычислить выражение $\sum_{i=0}^{10} \frac{i}{x^i}$

4. Написать метод, вычисляющий значение n/x^n . С его помощью вычислить выражение $\prod_{i=0}^{10} \frac{i}{x^i}$

5. Написать метод, который у четных чисел меняет знак, а нечетные числа оставляет без изменений. С его помощью обработать ряд чисел от 1 до 10. Результат вывести на экран.

6. Написать метод, который положительные числа возводит в квадрат, а отрицательные в куб. С его помощью обработать ряд чисел от -10 до 10. Результат вывести на экран.

7. Написать метод вычисления функции:

$$F = x^2 + x.$$

Организовать вычисление F для отрезка x от -10 до 10 с шагом 2.

8. Написать метод, вычисляющий значение $(n+1)/(x+1)^n$. С его помощью

вычислить выражение $\sum_{i=0}^{10} \frac{i+1}{(x+1)^i}$

9. Написать метод, вычисляющий значение $(n+1)/(x+1)^n$. С его помощью

вычислить выражение $\prod_{i=0}^{10} \frac{i+1}{(x+1)^i}$

10. Написать метод, который четные числа удваивает, а нечетные числа оставляет без изменений. С его помощью обработать ряд чисел от 1 до 10. Результат вывести на экран.

11. Написать метод, который положительные числа возводит в квадрат, а отрицательные в куб. С его помощью обработать ряд чисел от -10 до 10. Результат вывести на экран.

12. Написать метод, вычисляющий значение $(n+5)/(x+5)^n$. С его помощью

вычислить выражение $\sum_{i=0}^{10} \frac{i+5}{(x+5)^i}$

13. Написать метод, вычисляющий значение $(n+5)/(x+5)^n$. С его помощью

вычислить выражение $\prod_{i=0}^{10} \frac{i+5}{(x+5)^i}$

ЛАБОРАТОРНАЯ РАБОТА 6. СТРУКТУРЫ И КОЛЛЕКЦИИ

Задания к лабораторной работе 6

1. Выполнить задание согласно своему варианту.
2. Оформить созданное приложение: на форме должны быть отражены ФИО, все компоненты должны иметь подпись.

Методические указания к лабораторной работе 6

Цель: научиться использовать коллекции и структуры

Коллекция в языке C# — это объединение произвольного количества объектов, возможно, объектов разного типа.

Память под коллекцию не фиксируется, данные могут добавляться или удаляться по мере необходимости. Кроме того, для многих видов коллекций возможно включение в коллекцию данных разного типа.

Необобщенные коллекции предназначены для хранения элементов разного типа.

Создание коллекции:

```
ArrayList myList = new ArrayList();
```

Добавление элементов:

```
myList.Add("Computer");  
myList.Add("TV");  
myList.Add("House");
```

Вставка элемента перед индексом 1:

```
myList.Insert(1, "Hello");
```

Удаление элемента по индексу 2:

```
myList.RemoveAt(2);
```

Удаление элемента по значению "Hello":

```
myList.Remove("Hello");
```

Удаление всех элементов:

```
myList.Clear();
```

Обобщенные коллекции, предназначены для хранения элементов одного типа.

List (список) – это коллекция для хранения однотипных элементов.

Создание списка с элементами целого типа

```
List<int> myList = new List<int>();
```

Полностью поддерживаются методы коллекции ArrayList .

Структуры

Структуры – это составной объект, в который входят элементы любых типов, в том числе и функций. В отличие от массива, который является однородным объектом, структура может быть неоднородной, т.е. структура объединяет несколько переменных разного типа данных. Переменные, которые объединены структурой, называются членами, элементами или полями структуры.

```
struct имя
{
    Модификатор доступа    Тип переменной Имя переменной
    Модификатор доступа Тип возвращаемого значения Имя Метода
([Параметры])
}
```

Создание структуры без конструктора

```
public struct Book
{
    public string Name;
    public int Price;
}
```

Пример

```
Book book;
book.Name = «Про зайца»;
book.Price = 70;
```

Создание структуры с конструктором

```
public struct Book
{
    public string Name;
    public int Price;
    public Book(String name, int price)
    {
        Name = name;
        Price = price;
    }
}
```

```

    }
}

```

Пример: Book book = new Book (" Про зайца ", 70);

Пример выполнения задания

Автомобили				
Марка	Цвет	Цена	Изготовитель	Максимальная скорость
Определить суммарную стоимость всех автомобилей				

1. Добавить на форму элемент переключения вкладок TabControl.
2. На одной вкладке добавить текстовые поля для ввода данных для конкретной структуры согласно своему индивидуальному заданию.
3. На другой вкладке организовать возможность вывода данных, их редактирования и удаления выбранного элемента.

Рисунок 6 – Форма добавления элемента

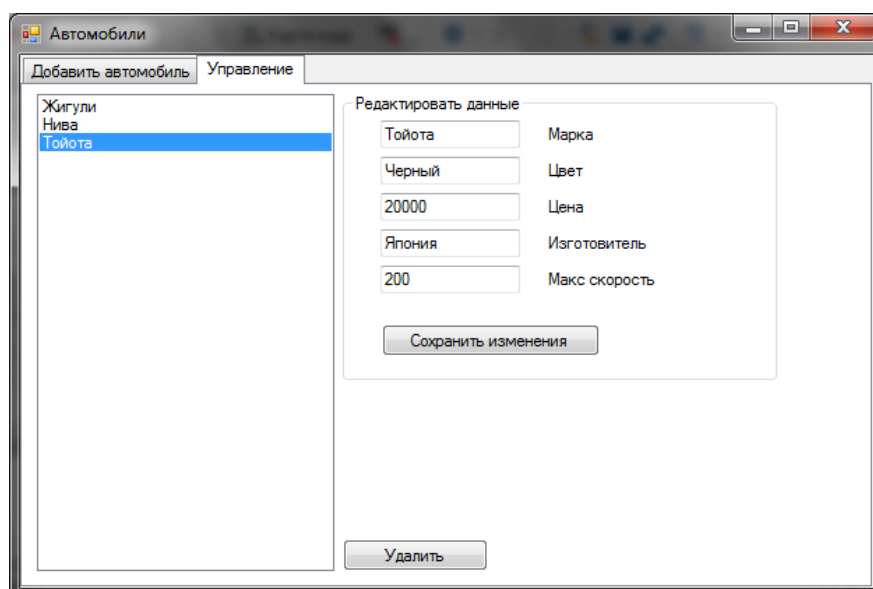


Рисунок 7 – Форма для редактирования данных

4. Определить структуру с полями согласно своему варианту.
5. Вывести на форму информацию согласно индивидуальному заданию.

Определение списка, состоящего из элементов типа структуры

```
List<car>allCar =new List<car>();
```

Добавление элементы структуры

```
allCar.add(new car (передача параметров в конструктор));
```

Удаление по индексу2:

```
allCar.RemoveAt(2);
```

Варианты индивидуальных заданий

1. Студенты факультета				
Номер зачетной книжки	Название группы	Фамилия, инициалы	Размер стипендии	Средний вступительный бал
Определить средний балл студентов факультета.				

2. Сотрудники отдела				
Табельный номер	Фамилия, инициалы	Дата рождения	Оклад в тыс. руб.	Стаж работы
Определить фонд заработной платы отдела.				

3. Города				
Название	Численность населения	Страна	Год создания	Площадь в кв. км
Определить город с максимальной численностью.				

4. Туристические маршруты агентства				
Наименование маршрута	Страна	Длительность тура	Стоимость	Количество человек
Определить суммарную стоимость туров.				

5. Самолеты				
Название типа	Фамилия конструктора	Год создания	Количество мест	Грузоподъемность в тоннах
Определить суммарную грузоподъемность самолетов.				
6. Перевозки				
Тип самолета	Номер борта	Количество рейсов	Налет в часах	Налет в тыс. км
Определить суммарный налет в часах.				

7. Расписание				
Номер рейса	Наименование рейса	Тип самолета	Стоимость билета	Время полета в часах
Определить минимальную и максимальную стоимость билетов.				

8. Экскурсии				
Наименование	Страна	Продолжительность	Стоимость	Транспорт
Определить самую продолжительную экскурсию.				

9. Спортивные секции				
Название секции	ФИО тренера	Длительность тренировки	Стоимость одного занятия	Количество спортсменов
Определить общее количество спортсменов по всем секциям.				

10. Музеи				
Название музея	Характер экспозиций	Адрес	Стоимость билета	Время работы
Определить минимальную и максимальную стоимость билетов.				

11. Кинотеатры				
Наименование кинотеатра	Адрес	Время сеансов	Стоимость билета	Количество мест
Определить минимальную и максимальную стоимость билетов.				

12. Линии метро				
Наименование	Район линии	Год пуска	Протяженность в км	Количество поездов
Определить общую протяженность линий метро.				

13. Книги				
Название книги	ФИО автора	Шифр	Издательство	Стоимость
Определить общую стоимость книг.				

ЛАБОРАТОРНАЯ РАБОТА 7. РАБОТА СО СТРОКАМИ

Задания к лабораторной работе 7

1. Выполнить задание согласно своему варианту.
2. Оформить созданное приложение: на форме должны быть отражены ФИО, все компоненты должны иметь подпись.

Методические указания к лабораторной работе 7

Цель: научиться работать со строками в языке C#

Методы класса **String**.

- **Compare** (Сравнивает два указанных объекта String с учетом регистра)

Метод возвращает число: <0 строка в классе раньше по алфавиту чем строка str =0 строка в классе одинакова строке str >0 значит строка в классе позже по алфавиту чем строка str

Пример

```
string str1 = "Hello";  
string str2 = "Hello";  
int result = str1.CompareTo(str2);  
// result = 0 // одинаковые строки
```

- **ToLower** (Возвращает копию этой строки, переведенную в нижний регистр)

Пример: C#

```
string str1 = "HELLO World!";  
string str2 = str1.ToLower();  
// str1 = "HELLO World!"  
// str2 = "hello world!"
```

- **ToUpper** (Возвращает копию этой строки, переведенную в верхний регистр)

Пример: C#

```
string str1 = "Hello World!";  
string str2 = str1.ToUpper();  
// str1 = "Hello World!"  
// str2 = "HELLO WORLD!"
```

- **Contains** (Возвращает значение, указывающее, встречается ли указанная подстрока внутри этой строки)

Метод возвращает bool:

true значит строка str встречается в строке класса

false значит строка str НЕ встречается в строке класса

Пример: C#

```
string str1 = "Hello world!";  
string str2 = "ello";  
bool bFound = str1.Contains(str2);  
// bFound = true
```

- **IndexOf** (Возвращает индекс с отсчетом от нуля. Возвращает -1, если знак или строка не найдена)

Пример: C#

```
string str1 = "Hello world!";  
string str2 = "lo";  
int pos = str1.IndexOf(str2);  
// pos = 3
```

- **Remove** (Возвращает новую строку, в которой удалено указанное число знаков текущей строки)

Если принимает один аргумент, то это позиция, начиная с которой обрезаются строка.

Если принимает два аргумента, то на первой позиции номер с которого необходимо обрезать, а на второй позиции количество обрезаемых символов

- **Substring()** Метод получения подстроки из строки.

Если принимает один аргумент – позиция, с которой будет начинаться новая подстрока.

Если принимает два аргумента, то на первой позиции номер с которого начнется подстрока, а на второй позиции длина подстроки.

Индивидуальные задания

1. Посчитать в строке количество слов.
2. Найти количество знаков препинания в исходной строке.

3. Дана строка символов. Вывести на экран цифры, содержащиеся в строке.
4. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести количество четных чисел в этой строке.
5. Дана строка символов. Вывести на экран количество строчных русских букв, входящих в эту строку.
6. Дана строка символов. Вывести на экран только строчные русские буквы, входящие в эту строку.
7. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Удалить первую букву в каждом слове.
8. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Заменить все буквы латинского алфавита на знак «+».
9. Дана строка символов, содержащая некоторый текст на русском языке. Заменить все большие буквы «А» на символ «*».
10. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Сформировать новую строку, состоящую из чисел длин слов в исходной строке.
11. Поменять местами первое и второе слово в исходной строке.
12. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести количество нечетных чисел в этой строке.
13. Посчитать в строке количество слов.

ЛАБОРАТОРНАЯ РАБОТА 8. ФАЙЛОВЫЙ ВВОД И ВЫВОД

Задания к лабораторной работе 8

1. Написать собственный текстовый редактор, используя файловый ввод и вывод.

Текстовый редактор должен позволять:

- Создавать новый текстовый документ;
- Открывать существующий текстовый документ;
- Сохранять текстовый документ.

2. Реализовать дополнительный функционал согласно своему варианту.

3. Оформить созданное приложение: на форме должны быть отражены ФИО, все компоненты должны иметь подпись.

Методические указания к лабораторной работе 8

Цель: освоить технологию файлового ввода и вывода.

Ход работы:

Создать новый проект

Добавить на форму следующие компоненты:

- MenuStrip (настроить согласно рисунку 8)
- ToolStrip
- RichTextBox
- OpenFileDialog
- SaveFileDialog.

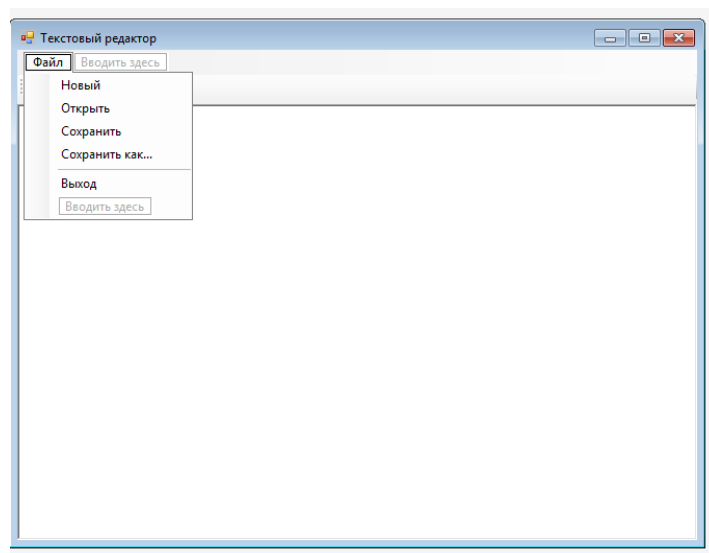


Рисунок 8 – Форма приложения

Назначение кнопок:

«Новый»

Создается новый тестовый документ с вызовом файлового диалога:

```
if (saveFileDialog1.ShowDialog() == DialogResult.OK)
{
    CurentFileNameFULL = saveFileDialog1.FileName;
    richTextBox1.SaveFile(CurentFileNameFULL,
RichTextBoxStreamType.RichText);
    richTextBox1.Clear();
}
```

«Открыть»

Открывается существующий текстовый файл

```
if (openFileDialog1.ShowDialog() == DialogResult.OK)
{
    CurentFileNameFULL = openFileDialog1.FileName;
    richTextBox1.LoadFile(CurentFileNameFULL,
RichTextBoxStreamType.RichText);
}
```

«Сохранить»

При нажатии на кнопку выполняется сохранение текущего файла.

```
richTextBox1.SaveFile(CurentFileNameFULL,  
RichTextBoxStreamType.RichText);
```

«Сохранить как...»

Выполняется открытие файлового диалога для сохранения файла.

```
if (saveFileDialog1.ShowDialog() == DialogResult.OK)  
{  
    CurentFileNameFULL = saveFileDialog1.FileName;  
    richTextBox1.SaveFile(CurentFileNameFULL,  
RichTextBoxStreamType.RichText);  
}
```

«Выход»

При нажатии на кнопку выполняется выход из программы.

Настройка компонентов

В свойстве Filter компонентов OpenFileDialog и SaveFileDialog установить значение "RTF-files"/*.rtf. Установка данного фильтра означает, что открываться и сохраняться файлы будут из формата *.rtf.

Компонент ToolStrip

Компонент позволяет создавать линейки инструментов с различными элементами пользовательского интерфейса.

Панель ToolStrip может содержать объекты следующих классов:

- ToolStripLabel: текстовая метка на панели инструментов, представляет функциональность элементов Label и LinkLabel.
- ToolStripButton: аналогичен элементу Button. Также имеет событие Click, с помощью которого можно обработать нажатие пользователя на кнопку.
- ToolStripSeparator: визуальный разделитель между другими элементами на панели инструментов.
- ToolStripToolStripComboBox: подобен стандартному элементу ComboBox.
- ToolStripTextBox: аналогичен текстовому полю TextBox.

Форматирование текста

Рассмотрим форматирование выделенного фрагмента текста на примере применения курсива.

Добавим на компоненте ToolStrip объект ToolStripLabel.

Для установки стиля шрифта к выделенному фрагменту используем синтаксис:

```
richTextBox1.SelectionFont = new  
Font(richTextBox1.SelectionFont, FontStyle.Italic);
```

`richTextBox1.SelectionFont` – стиль шрифта выделенного текста,

`new Font` – инициализация нового шрифта, в первом параметре задается стиль текущего выделения текста, а во втором параметре содержится информация о новом стиле. Если требуется одновременное применения нескольких шрифтов, то задается перечисление, например, для одновременного применения жирного стиля и курсива:

```
new Font(richTextBox1.SelectionFont, FontStyle.Bold |  
FontStyle.Italic);
```

Изменения цвета шрифта

Для выбора цвета используется компонент `ColorDialog`. Пример кода на событие выбора цвета текста:

```
if (colorDialog1.ShowDialog() == DialogResult.OK)  
{  
    richTextBox1.SelectionColor = colorDialog1.Color;  
}
```

Изменение размера шрифта

```
richTextBox1.SelectionFont = new  
Font(richTextBox1.SelectionFont.FontFamily, SizeFont,  
richTextBox1.SelectionFont.Style);
```

`SizeFont` – переменная целого типа, размер шрифта.

Индивидуальные задания

Создать возможность форматирование текста согласно своему варианту:

Вариант	Задание
1	Стиль жирный Установка цвета шрифта Выравнивание по центру Установка размера шрифта
2	Стиль курсив Установка цвета фона Выравнивание по левому краю Установка размера шрифта
3	Стиль подчеркнутый Установка цвета шрифта Выравнивание по правому краю Установка размера шрифта
4	Стиль жирный Установка цвета шрифта Выравнивание по центру Установка размера шрифта
5	Стиль курсив Установка цвета фона Выравнивание по левому краю Установка размера шрифта
6	Стиль подчеркнутый Установка цвета шрифта Выравнивание по правому краю Установка размера шрифта
7	Стиль жирный Установка цвета фона Выравнивание по центру Установка размера шрифта
8	Стиль курсив Установка цвета шрифта Выравнивание по левому краю Установка размера шрифта
9	Стиль подчеркнутый Установка цвета шрифта Выравнивание по правому краю Установка размера шрифта
10	Стиль жирный Установка цвета фона Выравнивание по центру Установка размера шрифта

11	Стиль курсив Установка цвета шрифта Выравнивание по правому краю Установка размера шрифта
12	Стиль подчеркнутый Установка цвета фона Выравнивание по левому краю Установка размера шрифта
13	Стиль жирный Установка цвета шрифта Выравнивание по центру Установка размера шрифта

ЛАБОРАТОРНАЯ РАБОТА 9. ФРАКТАЛЬНАЯ ГРАФИКА

Задания к лабораторной работе 9

1. Реализовать алгоритм рисования множества Мандельброта

Параметры, задаваемые на форме:

- ✓ Количество итераций для алгоритма
- ✓ Цветовые параметры R, G, B для функции выбора цвета

2. Оформить созданное приложение: на форме должны быть отражены ФИО, все компоненты должны иметь подпись.

Методические указания к лабораторной работе 9

Цель: знакомство с фрактальной графикой, реализация программы рисования алгебраических фракталов с применением компонента PictureBox.

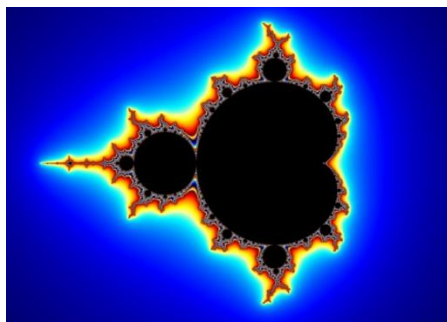


Рисунок 9 – Множество Мандельброта

Множество Мандельброта - это фрактал, определённый как множество точек c на комплексной плоскости, для которых итеративная последовательность

$$z_0 = 0$$

$$z_{n+1} = z_n^2 + c$$

не уходит на бесконечность.

Полное изображение множества Мандельброта строится на участке комплексной координатной плоскости от $x_0 = -2$ до $x_n = 1$ и от $y_0 = -1,5$ до $y_n = 1,5$.

Плоскость, на которой изображается множество Мандельброта имеет размеры: ширина W пикс., высота H пикс. (рис.10).

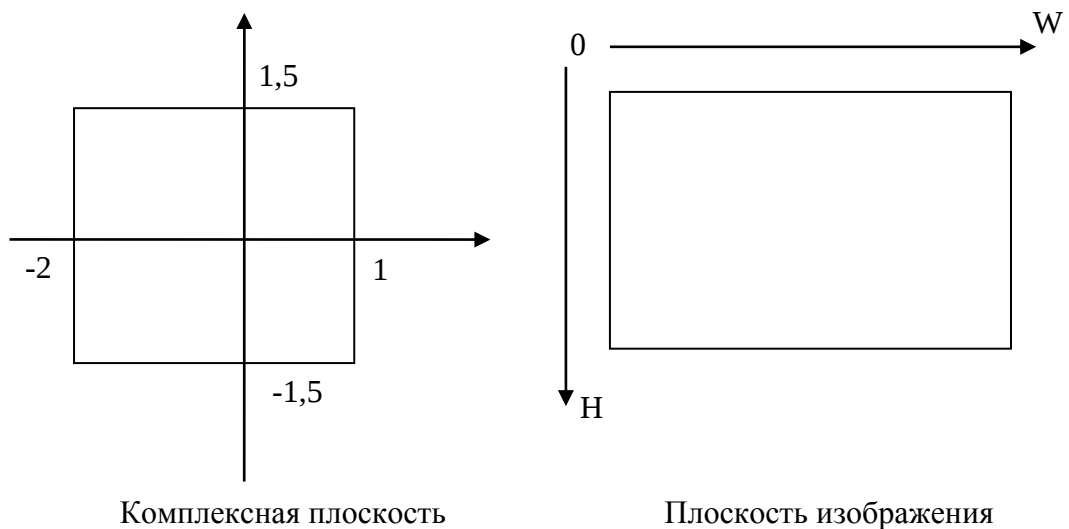


Рисунок 10 – Схема переноса координат

Чтобы осуществить переход от одной координатной плоскости к другой нужно взять точку плоскости (x, y) , на которой будут изображено множество, осуществить перенос этой точки на комплексную плоскость, сделать вычисления и отметить ее на исходной плоскости.

Точка комплексной плоскости имеет координаты:

$$c = x + iy$$

```
for (int x = 0; x < W; x++)
{
    cx = x0 + (xn - x0) / W * x; //координата x комплексного числа c
    for (int y = 0; y < H; y++)
        cy = y0 + (yn - y0) / H * y; //координата y комплексного
числа c
}
```

Таким образом, к каждой точке комплексной плоскости с координатами (cx, cy) применяется преобразование:

$$X_{i+1} = X_i^2 - Y_i^2 + cx$$

$$Y_{i+1} = 2 * X_i * Y_i + cy$$

$$X_0 = 0$$

$$Y_0 = 0$$

Окрашивание

Заранее определить, принадлежит ли точка (sx, sy) множеству Мандельброта нельзя. На каждом шаге, вычисляют расстояние точки с координатами (X, Y) от начала координат

$$r = \sqrt{X^2 + Y^2}$$

Если $\sqrt{X^2 + Y^2} < 2$ для всех итераций, значит точка принадлежит множеству Мандельброта и она окрашивается в **черный цвет**.

Если же $\sqrt{X^2 + Y^2} > 2$, точка окрашивается в цвет, зависящий от числа итераций, после которых точка удаляется на расстояние больше чем 2.

Рисование в Visual Studio

Создание рисунка bmp, он будет храниться в оперативной памяти

```
Bitmap img = new Bitmap(pictureBox1.Width, pictureBox1.Height);
```

Задание цвета пикселя

```
img.SetPixel(x, y, MyColor);
```

Способы задания цвета:

```
Color MyColor = Color.Black;
```

```
Color MyColor = Color.FromArgb(R, G, B);
```

где параметры R, G, B принимают значения от 0 до 255.

Запись изображения в PictureBox

```
pictureBox1.Image = img;
```

ЛАБОРАТОРНАЯ РАБОТА 10. ПРОСТЕЙШАЯ АНИМАЦИЯ

Задания к лабораторной работе 10

1. Реализовать движение по траектории согласно своему варианту.
2. Все кривые даны в полярной системе координат. Начальное значение угла $t = 0$, увеличение значения угла происходит на каждом такте таймера (обработчик Tick). Для перевода в декартову систему координат используется преобразование:

$$x = r \cos t$$

$$y = r \sin t$$

Методические указания к лабораторной работе 10

Цель: изучить возможности Visual Studio по созданию простейшей анимации. Написать и отладить программу, выводящую на экран анимационное изображение.

Работа с таймером

Класс для работы с таймером Timer формирует в приложении повторяющиеся события. События повторяются с периодичностью, указанной в миллисекундах в свойстве Interval. Установка свойства Enabled в значение true запускает таймер. Каждый тик таймера порождает событие Tick, обработчик которого обычно и создают в приложении.

Объект для рисования

Для рисования линий и фигур, отображения текста, вывода изображений и т. д. нужно использовать объект Graphics. Создание объекта Graphics:

```
Graphics g = pictureBox1.CreateGraphics();
```

Метод DrawLine рисует линию, соединяющую две точки с заданными координатами. У метода есть несколько перегруженных версий:

```
public void DrawLine(Pen, Point, Point);  
public void DrawLine(Pen, PointF, PointF);  
public void DrawLine(Pen, int, int, int, int);  
public void DrawLine(Pen, float, float, float, float);
```

Первый параметр задает инструмент для рисования линии – перо.

Перья создаются как объекты класса Pen, например:

```
Pen p = new Pen(Brushes.Black, 2);
```

Ход работы:

1. Добавить на форму компоненты PictureBox, Timer и Button.

2. В обработчике нажатия на кнопку:

- вычислить начальную точку для рисования

- запустить таймер:

```
timer1.Enabled = true;
```

3. В обработчике события таймера Tick написать код для пересчета очередных координат и их отрисовку:

```
g.DrawLine(pen, x1, y1, x2, y2);
```

где x1, y1 – значения, найденные для предыдущего угла t;

x2, y2 – значения, найденные для текущего угла t.

Индивидуальные задания

1. $r(t) = 1 + 7\cos(4t) + 4\sin^2(4t) + 3\sin^4(4t)$

2. $r(t) = 1 + \cos 5t - \sin^2 5t$

3. $r(t) = \sin(3/4t)$

4. $r(t) = \sin(5t) - \cos(10t)$

5. $r(t) = 1 + 7\cos(2t) + 4\sin^2(2t) + 3\sin^4(2t)$

6. $r(t) = 1 + \cos 2t - \sin^2 2t$

7. $r(t) = \sin(5t) - \cos(10t)$

8. $r(t) = (1 + \sin(t)) \cdot (1 + 0.9\cos(8t)) \cdot (1 + 0.1\cos(24t)) \cdot (0.5 + 0.05\cos(140t))$

9. $r(t) = 1 + 7\cos(4t) + 4\sin^2(4t) + 3\sin^4(4t)$

10. $r(t) = 1 + \cos 2t - \sin^2 2t$

11. $r(t) = \sin(5t) - \cos(10t)$

12. $r(t) = 1 + \cos 5t - \sin^2 5t$

13. $r(t) = 1 + \cos 2t - \sin^2 2t$

ЛАБОРАТОРНАЯ РАБОТА 11.

СОЗДАНИЕ ГРАФИЧЕСКОГО РЕДАКТОРА

Задания к лабораторной работе 11

1. Создайте приложение, реализующее простой графический редактор.
2. Функциями этого редактора должны быть:
 - открытие рисунка,
 - рисование поверх него простой кистью,
 - перевод изображения в градации серого,
 - сохранение рисунка в другой файл.
3. Оформить созданное приложение: на форме должны быть отражены ФИО, все компоненты должны иметь подпись.

Методические указания к лабораторной работе 11

Цель лабораторной работы: изучить возможности Visual Studio по открытию и сохранению файлов. Написать и отладить программу для обработки изображений.

Ход работы

Создайте форму и разместите на ней элементы управления Button и PictureBox.

В рассматриваемом случае не будем добавлять на форму элементы диалоговых окон OpenFileDialog и SaveFileDialog. Эти элементы будут порождены динамически в ходе выполнения программы с помощью конструктора. Например, так:

```
OpenFileDialog dialog = new OpenFileDialog();
```

Далее они будут вызываться с помощью метода ShowDialog().

Описание глобальных переменных

```
private Point PreviousPoint, point;  
private Bitmap bmp;  
private Pen blackPen;  
private Graphics g;
```

Обработчик события загрузки изображения

```

// Описываем объект класса OpenFileDialog
OpenFileDialog dialog = new OpenFileDialog();
// Задаем расширения файлов
dialog.Filter = "Image files (*.BMP, *.JPG, " +
    "*.GIF, *.PNG)|*.bmp;*.jpg;*.gif;*.png";
// Вызываем диалог и проверяем выбран ли файл
if (dialog.ShowDialog() == DialogResult.OK)
{
    // Загружаем изображение из выбранного файла
    Image image = Image.FromFile(dialog.FileName);
    int width = image.Width;
    int height = image.Height;
    pictureBox1.Width = width;
    pictureBox1.Height = height;
    // Создаем и загружаем изображение в формате bmp
    bmp = new Bitmap(image, width, height);
    // Записываем изображение в pictureBox1
    pictureBox1.Image = bmp;
    // Подготавливаем объект Graphics для рисования
    g = Graphics.FromImage(pictureBox1.Image);
}

```

Обработчик события загрузки формы

```

// Подготавливаем перо для рисования
blackPen = new Pen(Color.Black, 4);

```

Обработчик события нажатия мыши на PictureBox

```

// Записываем в предыдущую точку текущие координаты
PreviousPoint.X = e.X;
PreviousPoint.Y = e.Y;

```

Обработчик события перемещения мыши на PictureBox

```

// Проверяем нажата ли левая кнопка мыши
if (e.Button == MouseButton.Left)
{
    // Запоминаем текущее положение курсора мыши
    point.X = e.X;
    point.Y = e.Y;
    // Соединяем линией предыдущую точку с текущей
    g.DrawLine(blackPen, PreviousPoint, point);
    // Текущее положение курсора - в PreviousPoint
    PreviousPoint.X = point.X;
    PreviousPoint.Y = point.Y;
    // Принудительно вызываем перерисовку
    pictureBox1.Invalidate();
}

```

Обработчик события при сохранении файла

```

// Описываем и порождаем объект savedialog
SaveFileDialog savedialog = new SaveFileDialog();
// Задаем свойства для savedialog

```

```

savedialog.Title = "Сохранить картинку как ...";
savedialog.OverwritePrompt = true;
savedialog.CheckPathExists = true;
savedialog.Filter =
"Bitmap File (*.bmp)|*.bmp|" +
"GIF File (*.gif)|*.gif|" +
"JPEG File (*.jpg)|*.jpg|" +
"PNG File (*.png)|*.png";
// Показываем диалог и проверяем задано ли имя файла
if (savedialog.ShowDialog() == DialogResult.OK)
{
    string fileName = savedialog.FileName;
    // Убираем из имени расширение файла
    string strFileExtn = fileName.Remove(0, fileName.Length - 3);
    // Сохраняем файл в нужном формате
    switch (strFileExtn)
    {
        case "bmp":
            bmp.Save(fileName,
                System.Drawing.Imaging.ImageFormat.Bmp);
            break;
        case "jpg":
            bmp.Save(fileName,
                System.Drawing.Imaging.ImageFormat.Jpeg);
            break;
        case "gif":
            bmp.Save(fileName,
                System.Drawing.Imaging.ImageFormat.Gif);
            break;
        case "tif":
            bmp.Save(fileName,
                System.Drawing.Imaging.ImageFormat.Tiff);
            break;
        case "png":
            bmp.Save(fileName,
                System.Drawing.Imaging.ImageFormat.Png);
            break;
        default:
            break;
    }
}

```

Обработчик события нажатия мыши для перевода изображения в градации серого

```

// Циклы для перебора всех пикселей на изображении
for (int i = 0; i < bmp.Width; i++)
    for (int j = 0; j < bmp.Height; j++)
    {
        // Извлекаем в R значение красного цвета
        int R = bmp.GetPixel(i, j).R;
        // Извлекаем в G значение зеленого цвета
        int G = bmp.GetPixel(i, j).G;
        // Извлекаем в B значение синего цвета

```

```

        int B = bmp.GetPixel(i, j).B;
        // Вычисляем среднее арифметическое
        int Gray = (R + G + B) / 3;
        // Переводим число в значение цвета.
        // 255 – показывает степень прозрачности.
        // Остальные значения одинаковы
        Color p = Color.FromArgb(255, Gray, Gray, Gray);
        // Записываем цвет в текущую точку
        bmp.SetPixel(i, j, p);
    }
    // Вызываем функцию перерисовки окна
    Refresh();

```

Индивидуальные задания

Добавьте в приведенный графический редактор свои функции в соответствии с вариантом.

1. Создайте функцию, выводящую на изображение окружность. Центр окружности совпадает с центром изображения. Все точки внутри окружности переводятся в градации серого цвета. Все точки вне окружности остаются неизменными. Радиус окружности задается пользователем.

2. Разработайте функцию, добавляющую на изображение 1000 точек с координатами, заданными случайным образом. Цвет также задается случайным образом.

3. Создайте функцию, выводящую на изображение окружность. Центр окружности совпадает с центром изображения. Все точки вне окружности переводятся в градации серого цвета. Все точки внутри окружности остаются неизменными. Радиус окружности задается пользователем.

4. Создайте функцию, выводящую на изображение прямоугольник. Для всех точек внутри прямоугольника оставьте только канал B. Все точки вне прямоугольника переводятся в градации серого цвета.

5. Создайте функцию, разбивающую изображение на четыре равные части. В каждой оставьте значение только одного канала R, G и B, а в четвертой выведите градации серого цвета.

6. Разработайте функцию, которая каждую четную строку изображения переводит в градации серого цвета.

7. Разработайте функцию, которая переводит каждый нечетный столбец пикселей (вертикальные линии) в градации серого цвета.

8. Создайте функцию, разбивающую изображение на четыре равные части. В каждой оставьте значение только одного канала R, G и B, а в четвертой выведите градации серого цвета.

9. Создайте функцию, выводящую на изображение окружность. Центр окружности совпадает с центром изображения. Все точки внутри окружности переводятся в градации серого цвета. Все точки вне окружности остаются неизменными. Радиус окружности задается пользователем.

10. Создайте функцию, выводящую на изображение прямоугольник. Для всех точек внутри прямоугольника оставьте только канал G. Все точки вне прямоугольника переводятся в градации серого цвета.

11. Создайте функцию, разбивающую изображение на четыре равные части. В каждой оставьте значение только одного канала R, G и B, а в четвертой выведите градации серого цвета.

12. Разработайте функцию, добавляющую на изображение 1000 точек с координатами, заданными случайным образом. Цвет также задается случайным образом.

13. Создайте функцию, выводящую на изображение прямоугольник. Для всех точек вне прямоугольника оставьте только канал B. Все точки внутри прямоугольника переводятся в градации серого цвета.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

Основная литература

1. Гуриков, С. Р. Введение в программирование на языке Visual C# : учебное пособие / С.Р. Гуриков. — Москва : ФОРУМ : ИНФРА-М, 2020. — 447 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-00091-458-8. - Текст : электронный. – URL: <https://znanium.com/catalog/product/1092167>.

Дополнительная литература

2. Хорев, П. Б. Объектно-ориентированное программирование с примерами на C# : учебное пособие / П.Б. Хорев. — Москва : ФОРУМ : ИНФРА-М, 2020. — 200 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-00091-680-3. - Текст : электронный. - URL: <https://znanium.com/catalog/product/1069921>.

3. Шакин, В. Н. Объектно-ориентированное программирование на Visual Basic в среде Visual Studio .NET : учебное пособие / В. Н. Шакин, А. В. Загвоздкина, Г. К. Сосновиков. — Москва : ФОРУМ : ИНФРА-М, 2019. — 398 с. — (Высшее образование. Бакалавриат). - ISBN 978-5-00091-048-1. - Текст : электронный. - URL: <https://znanium.com/catalog/product/1010028>.

4. Смоленцев, Н. К. MATLAB: Программирование на Visual C#, Borland C#, JBuilder, VBA: Учебный курс : учебное пособие / Н. К. Смоленцев. — Москва : ДМК Пресс, 2008. — 464 с. — ISBN 978-5-388-00524-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/1253>.