Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Институт информационных технологий и анализа данных

наименование института

ОТЧЕТ  
к лабораторной работе по дисциплине

Технология программирования

Свой тип

Наименование темы

Выполнил студент группы ИСТб-19–2 Комогорцева Ю.В.

шифр Фамилия И.О.

Проверила Бахвалова З.А.

Фамилия И.О.

Содержание отчета на 34 стр.

Иркутск 2020 г.

### Постановка задачи

**Задача №9(оранжевая)**

1. Создать класс реализующий операции в соответствии с заданием
2. Протестировать операции
3. Создать GUI приложение, а-ля калькулятор

**Вектор**

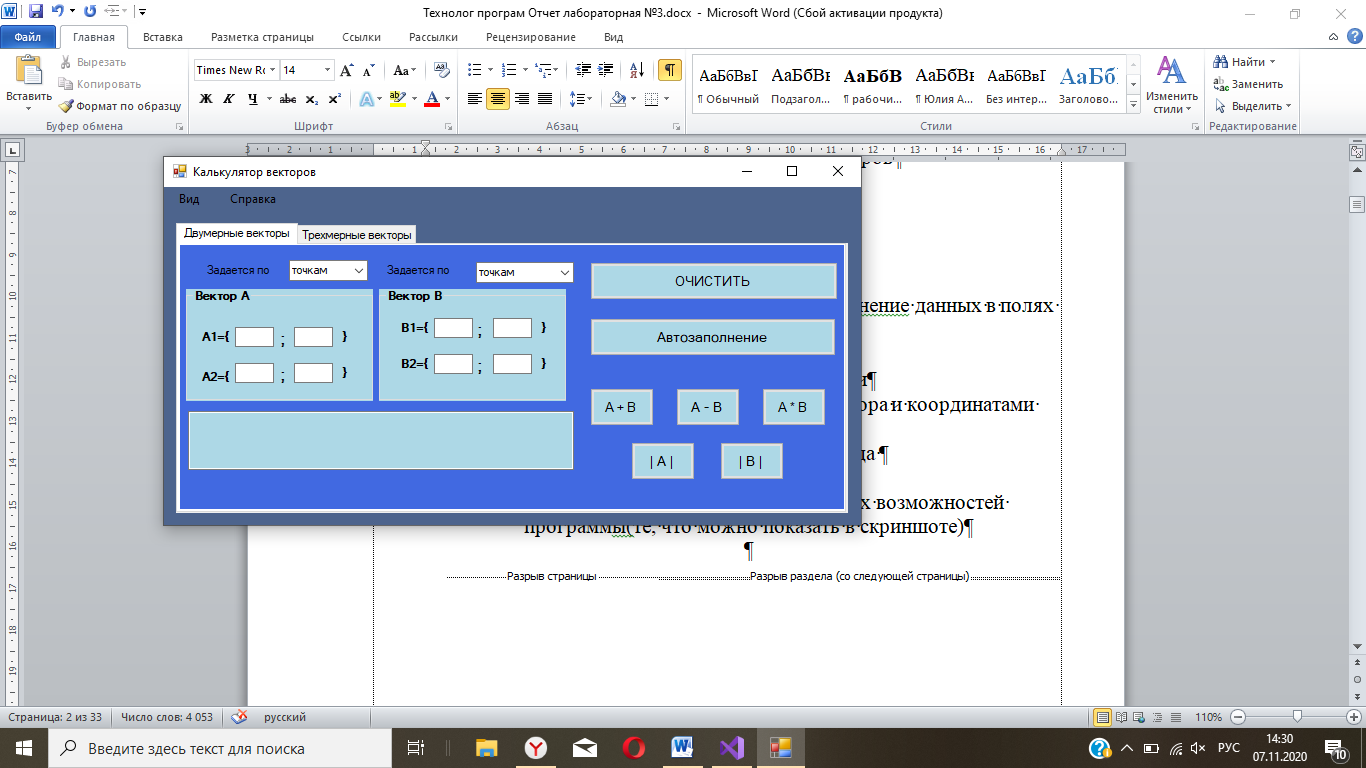
* сложение
* вычитание векторов
* вычисление скалярного произведения двух векторов
* длина вектора
* векторное произведение

**В программе реализовано:**

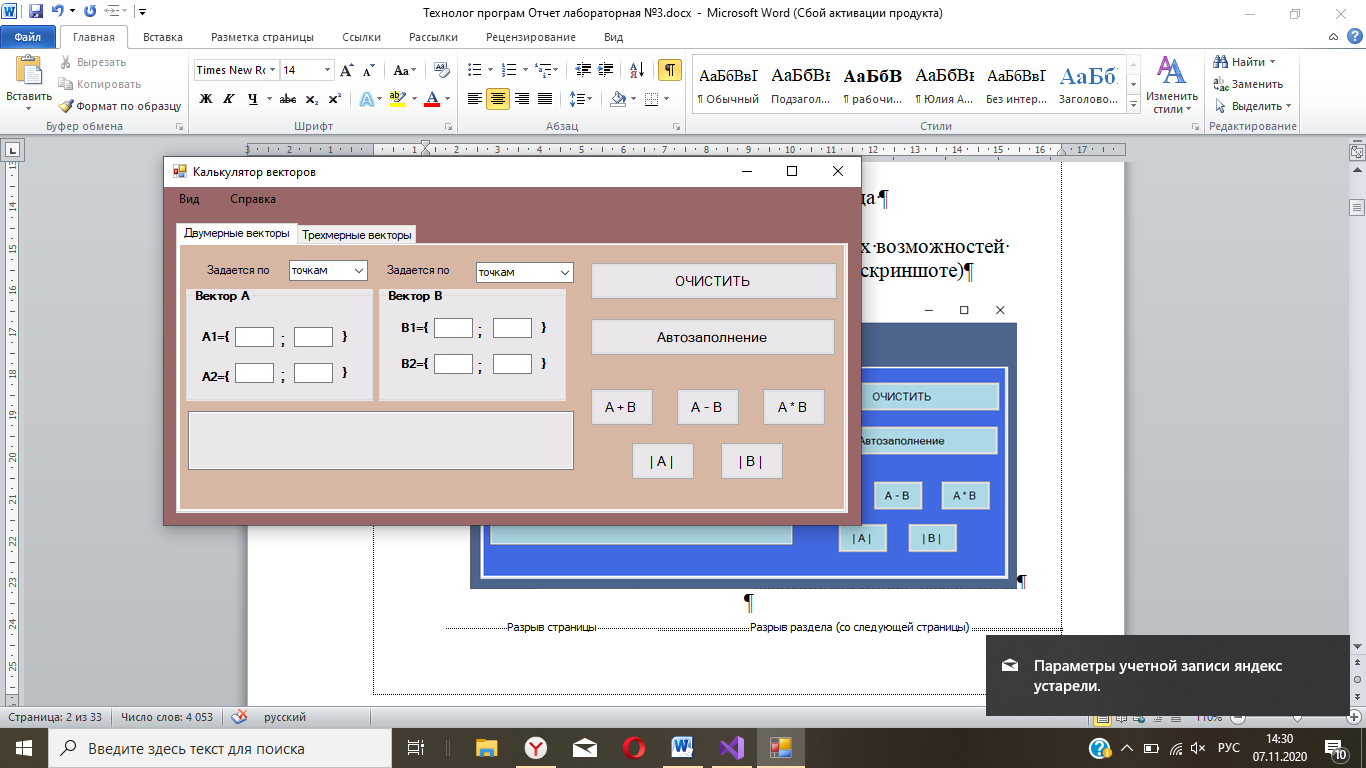
* Смена темы интерфейса
* Автоматический перерасчет результата при изменение данных в полях ввода
* «Справка»
* Работа с двухмерными и трехмерными векторами
* Возможность ввода векторов координатами вектора и координатами точек вектора
* Возможность автоматически заполнять поля ввода
* Возможность очистить поля ввода и результата

Ниже представлены скриншоты некоторых возможностей программы(те, что можно показать в скриншоте)

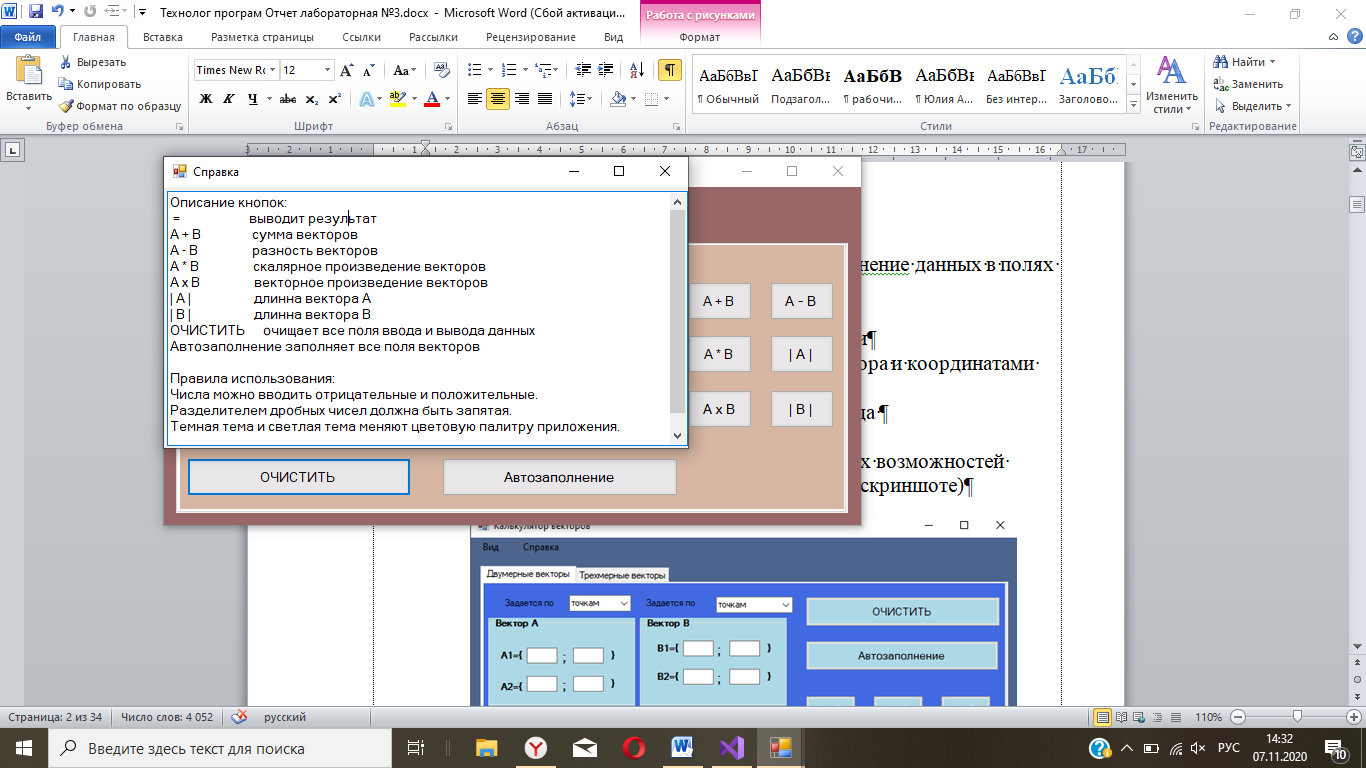
Темная тема

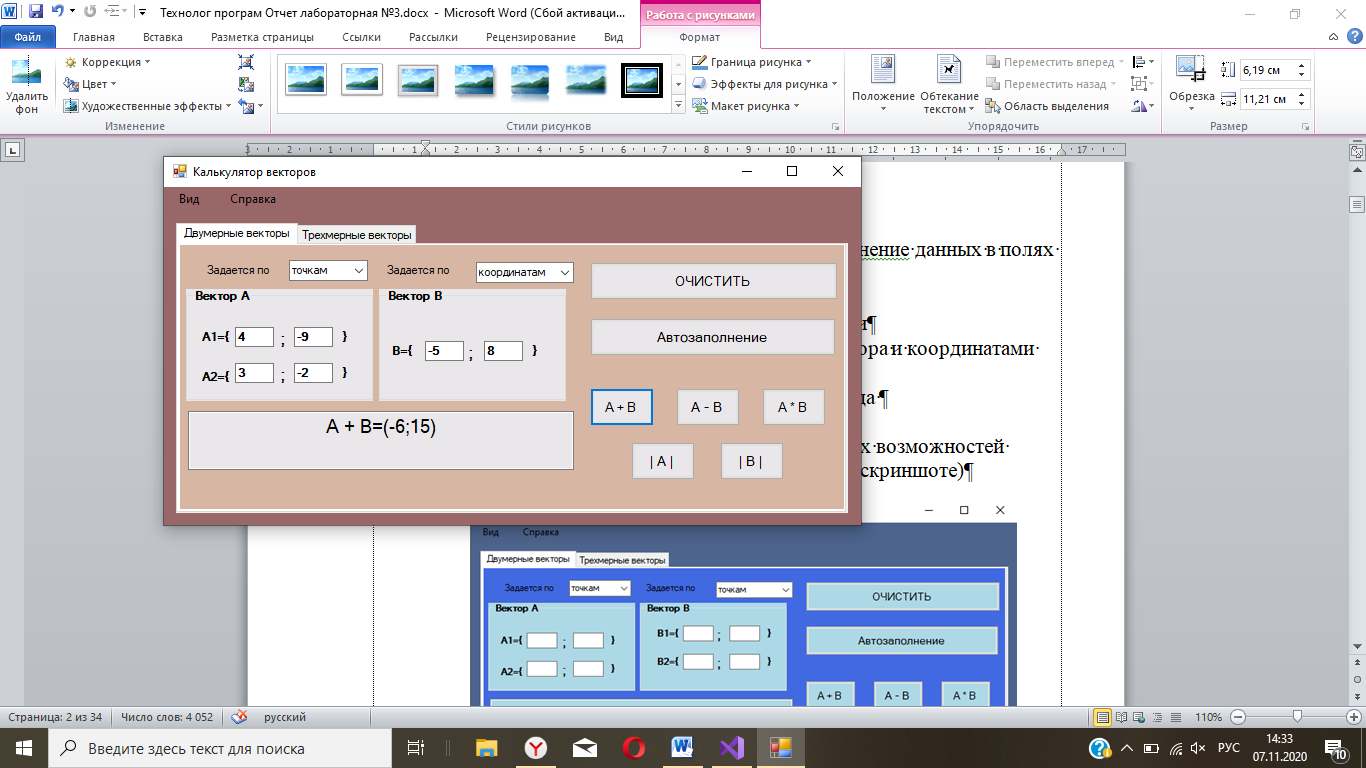


Светлая тема

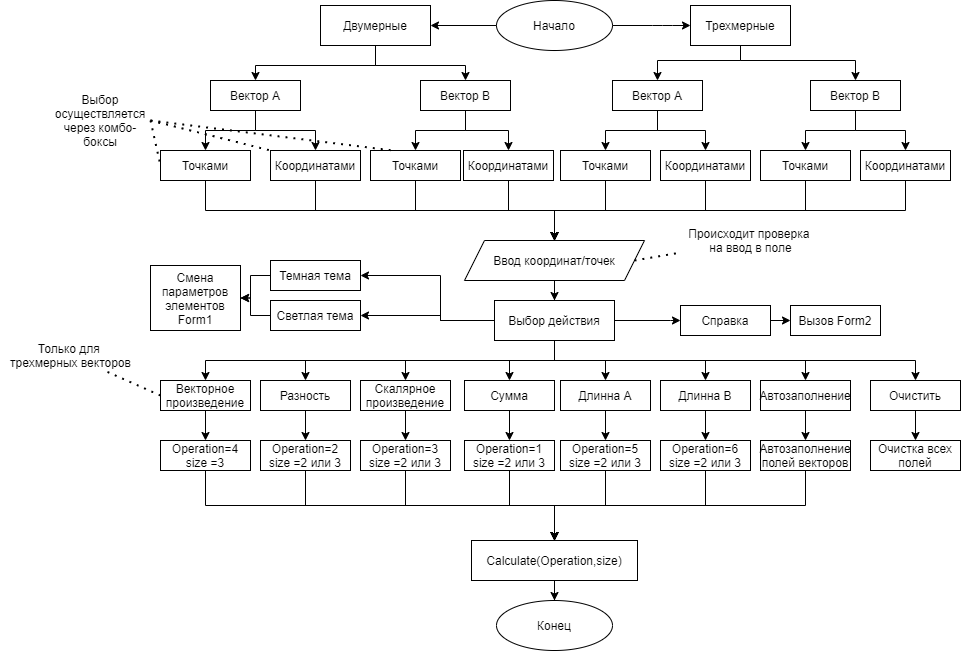


Справка

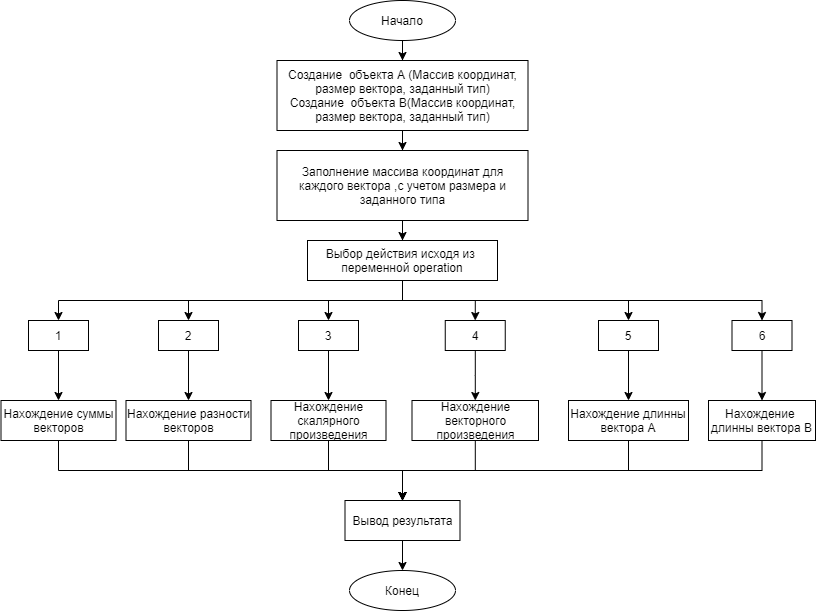


Возможность ввода векторов координатами вектора и координатами точек вектора 

**Схематическое описание работы программы**



**Схематическое описание работы метода Calculate()**



### Таблица описания классов,методов,конструкторов

### Класс Vector

***public Vector(double[] vec, MeasureVec size, TypeSetVec type)*** *-* конструктор,задает начальное значение для вектора(координаты,размер,тип)

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| Входные величины | | |
| vec | Размер матрицы | double[] |
| size | Матрицы для вычислений- объекты класса Matrix | MeasureVec |
| type | Итоговая матрица после вычислений | TypeSetVec |
| Выходные величины | | |
| vec | Размер матрицы | double[] |
| size | Матрицы для вычислений- объекты класса Matrix | MeasureVec |
| type | Итоговая матрица после вычислений | TypeSetVec |

***public int GetSize*** *–* свойство возвращающее размерность вектора, предназначено для удобства использования в циклах

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| *Выходные величины* | | |
| K | Число размерности | int |

***public static Vector To(Vector A )*** *-* метод для конвертации из заданного типа представления вектора "точки" в "координаты"

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| *Входные величины* | | |
| A | Вектор с полями (координаты,размер,тип) | *Vector* |
| *Выходные величины* | | |
| result | Вектор с полями (координаты,размер,тип) | *Vector* |

***public string Verbose()*** *-* свойство которое возвращает ответ в необходимом виде

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| *Выходные величины* | | |
| result | Строка с ответом | *string* |

***public static string Chislo(double rez)* -** метод позволяющий выводить результат не координатами ,а числом(нужен для длинны вектора и скалярного произведения)

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| *Входные величины* | | |
| rez | Число | *double* |
| *Выходные величины* | | |
| P | Строка состоящая из числа rez | *String* |

***public static Vector operator +(Vector A, Vector B)* -** оператор сложения векторов

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| *Входные величины* | | |
| A | Вектор с полями (координаты,размер,тип) | *Vector* |
| B | Вектор с полями (координаты,размер,тип) | *Vector* |
| *Выходные величины* | | |
| result | Вычисленный вектор с полями (координаты,размер,тип) | *Vector* |

***public static Vector operator -(Vector A, Vector B)*** – оператор вычитания векторов

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| *Входные величины* | | |
| A | Вектор с полями (координаты,размер,тип) | *Vector* |
| B | Вектор с полями (координаты,размер,тип) | *Vector* |
| *Выходные величины* | | |
| result | Вычисленный вектор с полями (координаты,размер,тип) | *Vector* |

***public static double operator \*(Vector A, Vector B)*** - оператор скалярного произведения векторов

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| *Входные величины* | | |
| A | Вектор с полями (координаты,размер,тип) | *Vector* |
| B | Вектор с полями (координаты,размер,тип) | *Vector* |
| *Выходные величины* | | |
| result | Вычисленное скалярное произведение | *Double* |

***public static Vector operator &(Vector A, Vector B)*** - оператор векторного произведения векторов

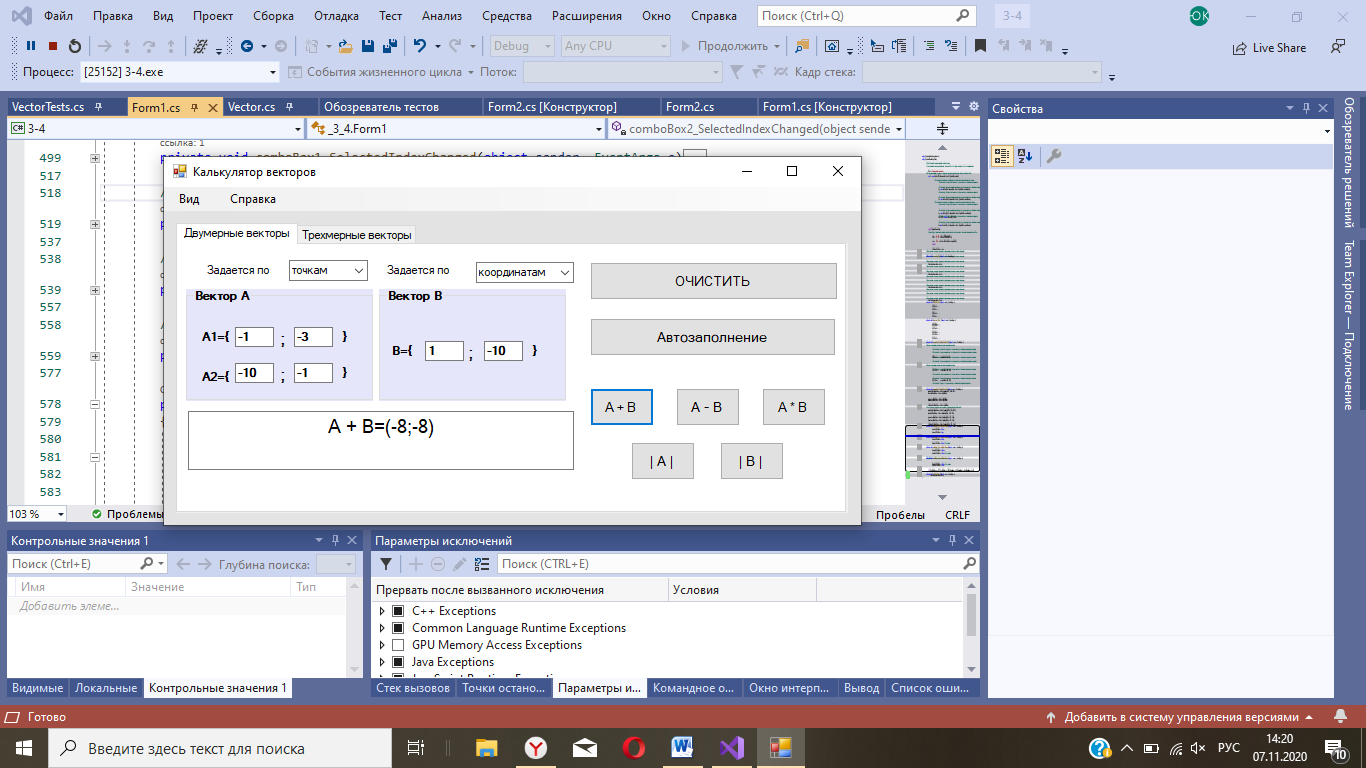
|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| *Входные величины* | | |
| A | Вектор с полями (координаты,размер,тип) | *Vector* |
| B | Вектор с полями (координаты,размер,тип) | *Vector* |
| *Выходные величины* | | |
| result | Вычисленный вектор с полями (координаты,размер,тип) | *Vector* |

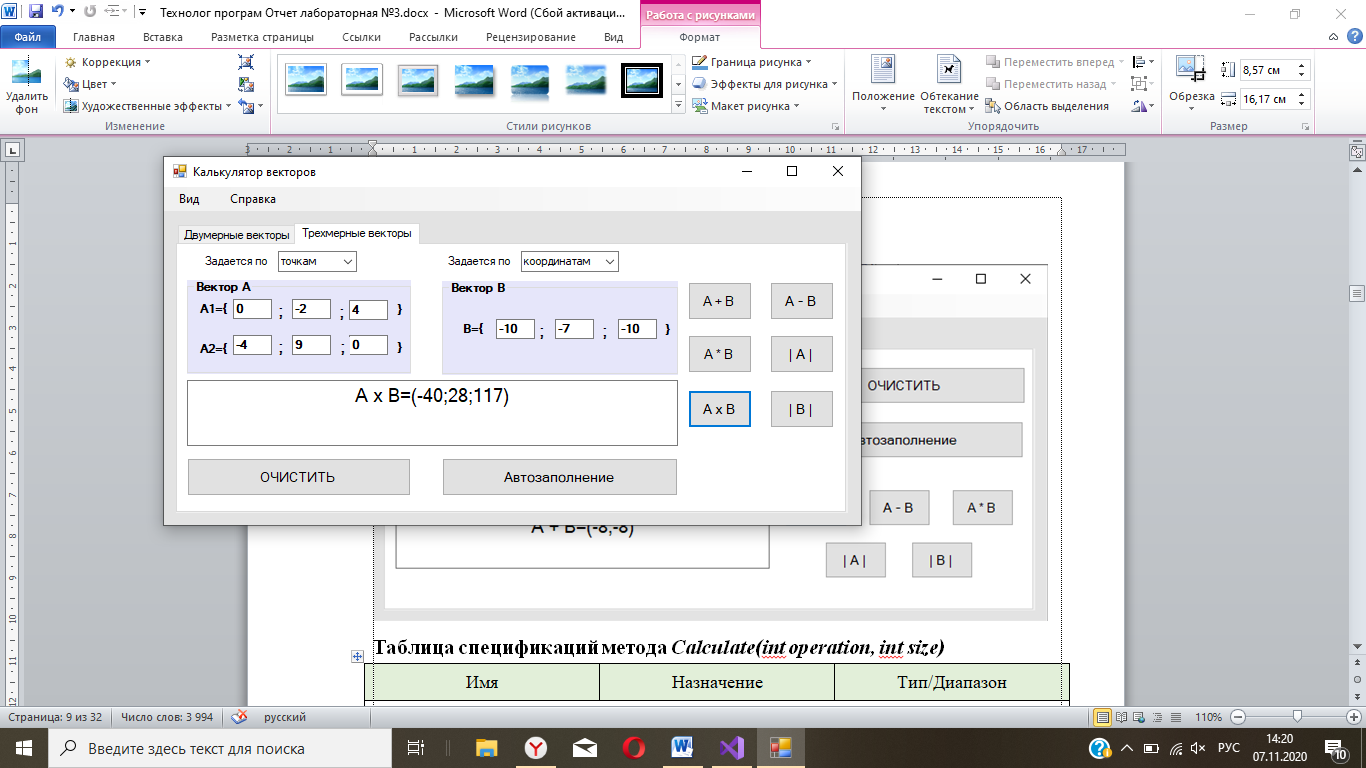
***public static double Lenght (Vector A)*** - метод нахождения длины вектора

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| *Входные величины* | | |
| A | Вектор с полями (координаты,размер,тип) | *Vector* |
| *Выходные величины* | | |
| result | Длинна вектора | Double |

### Form 1

Так выглядит Form1





### Таблица спецификаций метода *Calculate(int operation, int size)*

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| *Входные величины* | | |
| operation | Номер операции | int |
| size | Размер векторов | int |

**Методы для вычисления математических операций**

**над векторами, предназначенные для кнопок с этими операциями**

*private void сумма\_Click(object sender, EventArgs e)*

*private void разность\_Click(object sender, EventArgs e)*

*private void Скаляр\_Click(object sender, EventArgs e)*

*private void длиннаА\_Click(object sender, EventArgs e)*

*private void длиннаB\_Click(object sender, EventArgs e)*

*private void Сумма2\_Click(object sender, EventArgs e)*

*private void Скаляр2\_Click(object sender, EventArgs e)*

*private void Длинна\_а\_Click(object sender, EventArgs e)*

*private void Длинна\_в\_Click(object sender, EventArgs e)*

*private void Разн\_Click(object sender, EventArgs e)*

*private void Векторн\_Click(object sender, EventArgs e)*

**Методы для очистки всех полей , предназначенные для этих кнопок**

*private void ОЧИСТИТЬ\_Click(object sender, EventArgs e)*

*private void Очистить\_Click\_1(object sender, EventArgs e)*

**Методы для автоматического заполнения всех полей**

*private void auto1\_Click(object sender, EventArgs e)*

*private void auto3\_Click(object sender, EventArgs e)*

**Метод для вызова справки(Form2)**

*private void справкаToolStripMenuItem1\_Click(object sender, EventArgs e)*

**Методы для изменения цветов формы**

*private void темныйToolStripMenuItem\_Click(object sender, EventArgs e)*

*private void светлая\_Click(object sender, EventArgs e)*

**Методы для выбора заданного типа векторов**

*private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)*

*private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)*

*private void comboBox3\_SelectedIndexChanged(object sender, EventArgs e)*

*private void comboBox4\_SelectedIndexChanged(object sender, EventArgs e)*

**Метод для проверки вводимого значения**

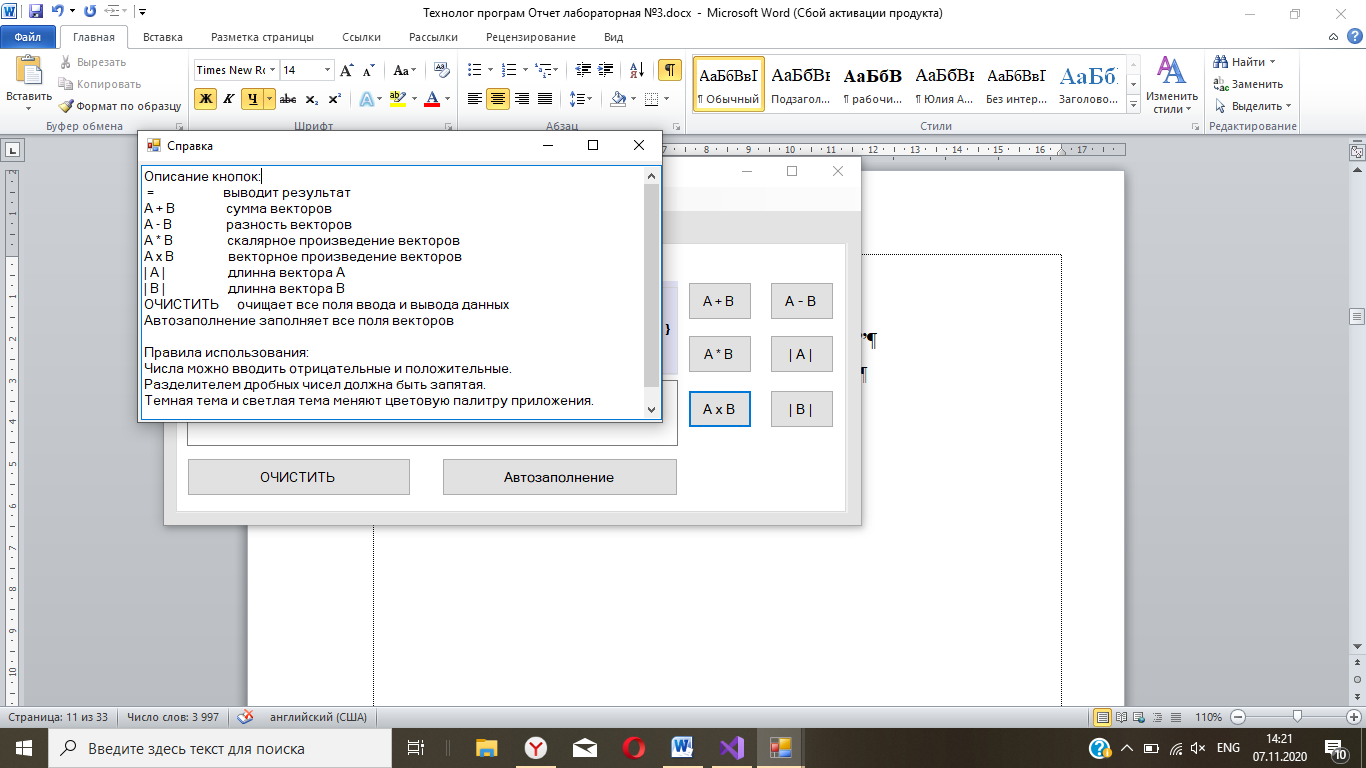
*public void proverka(object sender, KeyPressEventArgs e)-*проверка ввода –можно вводить только цифры, запятую,знак минус(отрицательное число) и Backspace(для удаления и дальнейшего ввода)

**Метод для автоматического перерасчета рез**у**льтата при изменение полей векторов**

*private void onValueChanged(object sender, EventArgs e)*

**Form 2**

Так выглядит Form2



**Для бокса с информацией “справка”**

*private void textBox1\_TextChanged\_1(object sender, EventArgs e)*

### Таблица тестов

| Номер теста | Что проверяем | Входные данные | Выходные данные |
| --- | --- | --- | --- |
| **Автоматизированное тестирование для класса Vector** | | | |
| 1 | Проверка правильной работы Verbose()-возвращает ответ в необходимом виде | new Vector ({ 1, 2 }, MeasureVec.size2, TypeSetVec.coordinate) | (1;2) |
| 2 | new Vector(new double[] { 1, 2, 3 }, MeasureVec.size3, TypeSetVec.coordinate) | (1;2;3) |
| 3 | Проверка правильной работы метода To(Vector A)-конвертирует из координат точек A и В в координаты вектора AB | new Vector(new double[] { 2, -4, 4, 4 }, MeasureVec.size2, TypeSetVec.point) | (2;8) |
| 4 | new Vector(new double[] { 1, 2, 3, 4, 5, 6 }, MeasureVec.size3, TypeSetVec.point) | (3;3;3) |
| 5 | Проверка правильной работы оператора (A + B) –сумма векторов | A = new Vector(new double[] { 1, 2, 3, 4 }, MeasureVec.size2, TypeSetVec.point)  B = new Vector(new double[] { 1, 0, 2, 2 }, MeasureVec.size2, TypeSetVec.point) | (3;4) |
| 6 | Проверка правильной работы оператора (A - B) –разность векторов | A = new Vector(new double[] { 1, 2, 3, 4 }, MeasureVec.size2, TypeSetVec.point)  B = new Vector(new double[] { 1, 0, 2, 2 }, MeasureVec.size2, TypeSetVec.point) | (1;0) |
| 7 | Проверка правильной работы оператора (A \* B) –скалярное произведение векторов | A = new Vector(new double[] { 1, 2, 3, 4 }, MeasureVec.size2, TypeSetVec.point)  B = new Vector(new double[] { 1, 0, 2, 2 }, MeasureVec.size2, TypeSetVec.point) | 6 |

Продолжение таблицы тестов

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | Проверка правильной работы оператора (A &B) –векторное произведение векторов | A = new Vector(new double[] { -1, 2, 5, 6, 4, 0 }, MeasureVec.size3, TypeSetVec.point)  B = new Vector(new double[] { 1, 2, 3, -2, 4, 6 }, MeasureVec.size3, TypeSetVec.point) | (16;-6;20) |
| 9 | A = new Vector(new double[] { 4, 5, 6}, MeasureVec.size3, TypeSetVec.coordinate);  B = new Vector(new double[] { 1, 2, -3}, MeasureVec.size3, TypeSetVec.coordinate); | (-27;18;3) |
| 10 | A = new Vector(new double[] { 4, 5, 6 }, MeasureVec.size3, TypeSetVec.coordinate);  B = new Vector(new double[] { 1, 2, 3, -2, 4, 6 }, MeasureVec.size3, TypeSetVec.point); | (3;-30;23) |
| 11 | Проверка правильной работы метода Lenght ()-находит длинну входящего вектора | A = new Vector(new double[] { 1, 2, 3, 4 }, MeasureVec.size2, TypeSetVec.point) | 2.83 |
| 12 | B = new Vector(new double[] { 2, -4, 4,4 }, MeasureVec.size2, TypeSetVec.point) | 8.25 |

### 

### Код программы

**Класс Vector**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3\_4

{

//предназначено для записи размера:двумерные и трехмерные векторы

public enum MeasureVec { size2, size3 };

//предназначено для записи заданного типа:точками и координатами

public enum TypeSetVec { point, coordinate };

public class Vector

{

private double[] vec; //для координат и точек векторов

private MeasureVec size; //для размерности вектора

private TypeSetVec type; //для заданного типа

//конструктор,задает начальное значение для вектора

public Vector(double[] vec, MeasureVec size, TypeSetVec type)

{

this.vec = vec;

this.type = type;

this.size = size;

}

// свойство возвращающее размерность вектора,предназначено для удобства использования в циклах

public int GetSize

{

get {

int k=0;

switch (this.size)

{

case MeasureVec.size2:

k = 2;

break;

case MeasureVec.size3:

k = 3;

break;

}

return k;

}

}

//метод для конвертации из заданного типа представления вектора "точки" в "координаты"

public static Vector To(Vector A)

{

Vector result = new Vector(A.vec, A.size, TypeSetVec.coordinate);

if (A.type == TypeSetVec.point)

{

for (int i = 0; i < A.GetSize; i++)

{

result.vec[i] = A.vec[A.GetSize + i] - A.vec[i];

}

}

return result;

}

//свойство которое возвращает ответ в необходимом виде

public string Verbose()

{

int k=0;

switch(this.size)

{

case MeasureVec.size2:

k = 2;

break;

case MeasureVec.size3:

k = 3;

break;

}

string s = " ";

if (k == 2)

{

s = "(" + vec[0] + ";" + vec[1] + ")";

}

else

{

s = "(" + vec[0] + ";" + vec[1] + ";" + vec[2] + ")";

}

return s;

}

//метод позволяющий выводить результат не координатами ,а числом

public static string Chislo(double rez)

{

string P = string.Format("{0}", rez);

return P;

}

//оператор сложения векторов

public static Vector operator +(Vector A, Vector B)

{

//векторы A и B конвертируем в задаваемый тип "координаты"

A =To(A);

B =To(B);

//создаем новый обьект Vector

Vector result = new Vector(A.vec,A.size, A.type);

//цикл сложения координат векторов

for (int i = 0; i < A.GetSize ; i++) {

result.vec[i]=A.vec[i] + B.vec[i];

}

return result;

}

//оператор вычитания векторов

public static Vector operator -(Vector A, Vector B)

{

//векторы A и B конвертируем в задаваемый тип "координаты"

A = To(A);

B=To(B);

//создаем новый обьект Vector

Vector result = new Vector(A.vec, A.size, A.type);

//цикл вычитания координат векторов

for (int i = 0; i < A.GetSize; i++)

{

result.vec[i] = A.vec[i] - B.vec[i];

}

return result;

}

//оператор скалярного произведения векторов

public static double operator \*(Vector A, Vector B)

{

double result=0 ;

//векторы A и B конвертируем в задаваемый тип "координаты"

A = To(A);

B = To(B);

//цикл нахождения скалярного произведения векторов(на выходе число)

for (int i = 0; i < A.GetSize; i++)

{

result = result+ (A.vec[i] \* B.vec[i]);

}

return result;

}

//оператор векторного произведения векторов

public static Vector operator &(Vector A, Vector B)

{

//векторы A и B конвертируем в задаваемый тип "координаты"

A = To(A);

B = To(B);

//создаем новый обьект Vector

Vector result = new Vector(A.vec, A.size, A.type);

double[] r = new double[A.GetSize];

r[0] = A.vec[1] \* B.vec[2] - A.vec[2] \* B.vec[1];

r[1] = A.vec[2] \* B.vec[0]- A.vec[0] \* B.vec[2];

r[2] = A.vec[0] \* B.vec[1] - A.vec[1] \* B.vec[0];

for (int i = 0; i < A.GetSize; i++)

{

result.vec[i] = r[i];

}

return result;

}

//метод нахождения длнны вектора

public static double Lenght (Vector A)

{

double result = 0;

A = To(A);

for (int i = 0; i < A.GetSize; i++)

{

result += Math.Pow(A.vec[i], 2);

}

result=Math.Sqrt(result);

result = Math.Round(result, 2);

return result;

}

}

}

**Код автоматизированного тестирования класса Vector**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using \_3\_4;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3\_4.Tests

{

[TestClass()]

public class VectorTests

{

[TestMethod()]

public void VerboseTest()

{

var A = new Vector(new double[] { 1, 2 }, MeasureVec.size2, TypeSetVec.coordinate);

Assert.AreEqual("(1;2)", A.Verbose());

A = new Vector(new double[] { 1, 2, 3 }, MeasureVec.size3, TypeSetVec.coordinate);

Assert.AreEqual("(1;2;3)", A.Verbose());

}

[TestMethod()]

public void ToTest()

{

var A = new Vector(new double[] { 2, -4, 4, 4 }, MeasureVec.size2, TypeSetVec.point);

Assert.AreEqual("(2;8)", Vector.To(A).Verbose());

A = new Vector(new double[] { 1, 2, 3, 4, 5, 6 }, MeasureVec.size3, TypeSetVec.point);

Assert.AreEqual("(3;3;3)", Vector.To(A).Verbose());

}

[TestMethod()]

public void SumTest()

{

var A = new Vector(new double[] { 1, 2, 3, 4 }, MeasureVec.size2, TypeSetVec.point);//AB(2,2)

var B = new Vector(new double[] { 1, 0, 2, 2 }, MeasureVec.size2, TypeSetVec.point);//AB(1,2)

Assert.AreEqual("(3;4)", (A + B).Verbose());

}

[TestMethod()]

public void RazTest()

{

var A = new Vector(new double[] { 1, 2, 3, 4 }, MeasureVec.size2, TypeSetVec.point);//AB(2,2)

var B = new Vector(new double[] { 1, 0, 2, 2 }, MeasureVec.size2, TypeSetVec.point);//AB(1,2)

Assert.AreEqual("(1;0)", (A - B).Verbose());

}

[TestMethod()]

public void ScalTest()

{

var A = new Vector(new double[] { 1, 2, 3, 4 }, MeasureVec.size2, TypeSetVec.point);//AB(2,2)

var B = new Vector(new double[] { 1, 0, 2, 2 }, MeasureVec.size2, TypeSetVec.point);//AB(1,2)

double c = 6;

Assert.AreEqual(c, A\*B);

}

[TestMethod()]

public void VecTest()

{

var A = new Vector(new double[] { -1, 2, 5, 6, 4, 0 }, MeasureVec.size3, TypeSetVec.point);//AB(3,3,3)

var B = new Vector(new double[] { 1, 2, 3, -2, 4, 6 }, MeasureVec.size3, TypeSetVec.point);//AB(1,2,3)

Assert.AreEqual("(16;-6;20)", (A & B).Verbose());

A = new Vector(new double[] { 4, 5, 6}, MeasureVec.size3, TypeSetVec.coordinate);//AB(3,3,3)

B = new Vector(new double[] { 1, 2, -3}, MeasureVec.size3, TypeSetVec.coordinate);//AB(1,2,3)

Assert.AreEqual("(-27;18;3)", (A & B).Verbose());

A = new Vector(new double[] { 4, 5, 6 }, MeasureVec.size3, TypeSetVec.coordinate);//AB(3,3,3)

B = new Vector(new double[] { 1, 2, 3, -2, 4, 6 }, MeasureVec.size3, TypeSetVec.point);//AB(1,2,3)

Assert.AreEqual("(3;-30;23)", (A & B).Verbose());

}

[TestMethod()]

public void lenghtTest()

{

var A = new Vector(new double[] { 1, 2, 3, 4 }, MeasureVec.size2, TypeSetVec.point);//AB(2,2)

var B = new Vector(new double[] { 2, -4, 4,4 }, MeasureVec.size2, TypeSetVec.point);//AB(1,2)

double rez= 2.83 ;

Assert.AreEqual(rez, Vector.Lenght(A));

rez = 8.25;

Assert.AreEqual(rez, Vector.Lenght(B));

}

}

}

**Код Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace \_3\_4

{

public partial class Form1 : Form

{

//что бы хранить какую операцию сейчас делаем

public int operation=0;

//что бы хранить размер вектора

public int size;

// для хранения типа представления вектора.false это задано точками, true это координатами

public bool typeA;

// для хранения типа представления вектора.false это задано точками, true это координатами

public bool typeB;

public Form1()

{

InitializeComponent();

Text = "Калькулятор векторов";

}

//метод выполняющий создание и заполение векторов,математические подчеты и вывод ответа

private void Calculate(int operation, int size)

{

double[] V = new double[0];

double[] V1 = new double[0];

var A = new Vector(V, MeasureVec.size2, TypeSetVec.point);

var B = new Vector(V1, MeasureVec.size2, TypeSetVec.point);

try

{

//происходит выбор по размерности вектора

switch (size)

{

case 2:

//если векторы двумерные,то выбираем по каким типам представления они заданы

switch (typeA)

{

case false:

//если вектор А задан 2мя точками то создаем вектор и записываем координаты

double[] Vec = new double[4];

Vec[0] = double.Parse(A1X.Text);

Vec[1] = double.Parse(A1Y.Text);

Vec[2] = double.Parse(A2X.Text);

Vec[3] = double.Parse(A2Y.Text);

A = new Vector(Vec, MeasureVec.size2, TypeSetVec.point);

break;

case true:

//если вектор А задан координатами вектора то создаем вектор и записываем координаты

Vec = new double[2];

Vec[0] = double.Parse(AX.Text);

Vec[1] = double.Parse(AY.Text);

A = new Vector(Vec, MeasureVec.size2, TypeSetVec.coordinate);

break;

}

switch (typeB)

{

case false:

//если вектор В задан 2мя точками то создаем вектор и записываем координаты

double[] Vec1 = new double[4];

Vec1[0] = double.Parse(B1X.Text);

Vec1[1] = double.Parse(B1Y.Text);

Vec1[2] = double.Parse(B2X.Text);

Vec1[3] = double.Parse(B2Y.Text);

B = new Vector(Vec1, MeasureVec.size2, TypeSetVec.point);

break;

case true:

//если вектор В задан координатами вектора то создаем вектор и записываем координаты

Vec1 = new double[2];

Vec1[0] = double.Parse(BX.Text);

Vec1[1] = double.Parse(BY.Text);

B = new Vector(Vec1, MeasureVec.size2, TypeSetVec.coordinate);

break;

}

break;

case 3:

//если векторы трехмерные,то выбираем по каким типам представления они заданы

switch (typeA)

{

case false:

//если вектор А задан 2мя точками то создаем вектор и записываем координаты

double[] Vec\_0 = new double[6];

Vec\_0[0] = double.Parse(A1\_X.Text);

Vec\_0[1] = double.Parse(A1\_Y.Text);

Vec\_0[2] = double.Parse(A1\_Z.Text);

Vec\_0[3] = double.Parse(A2\_X.Text);

Vec\_0[4] = double.Parse(A2\_Y.Text);

Vec\_0[5] = double.Parse(A2\_Z.Text);

A = new Vector(Vec\_0, MeasureVec.size3, TypeSetVec.point);

break;

case true:

//если вектор А задан координатами вектора то создаем вектор и записываем координаты

Vec\_0 = new double[3];

Vec\_0[0] = double.Parse(A\_X.Text);

Vec\_0[1] = double.Parse(A\_Y.Text);

Vec\_0[1] = double.Parse(A\_Z.Text);

A = new Vector(Vec\_0, MeasureVec.size3, TypeSetVec.coordinate);

break;

}

switch (typeB)

{

case false:

//если вектор В задан 2мя точками то создаем вектор и записываем координаты

double[] Vec\_1 = new double[6];

Vec\_1[0] = double.Parse(B1\_X.Text);

Vec\_1[1] = double.Parse(B1\_Y.Text);

Vec\_1[2] = double.Parse(B1\_Z.Text);

Vec\_1[3] = double.Parse(B2\_X.Text);

Vec\_1[4] = double.Parse(B2\_Y.Text);

Vec\_1[5] = double.Parse(B2\_Z.Text);

B = new Vector(Vec\_1, MeasureVec.size3, TypeSetVec.point);

break;

case true:

//если вектор В задан координатами вектора то создаем вектор и записываем координаты

Vec\_1 = new double[3];

Vec\_1[0] = double.Parse(B\_X.Text);

Vec\_1[1] = double.Parse(B\_Y.Text);

Vec\_1[1] = double.Parse(B\_Z.Text);

B= new Vector(Vec\_1, MeasureVec.size3, TypeSetVec.coordinate);

break;

}

break;

default:

break;

}

}

catch (FormatException)

{

}

string rez = "";

//далее будет происходить решение математической части,исходя из значения переменной operation

switch (operation)

{

case 1:

rez ="A + B="+(A + B).Verbose();

break;

case 2:

rez = "A - B=" + (A - B).Verbose();

break;

case 3:

rez = "A \* B=" + Vector.Chislo(A \* B);

break;

case 4:

rez = "A x B=" + (A & B).Verbose();

break;

case 5:

rez = "|A|=" + Vector.Chislo(Vector.Lenght(A));

break;

case 6:

rez = "|B|=" + Vector.Chislo(Vector.Lenght(B));

break;

default:

break;

}

switch (size)

{

case 2:

txtResult.Text =rez;

break;

case 3:

txtResult2.Text = rez;

break;

}

}

//при нажатии на кнопку произойдет присваивание размера и номера операции

private void сумма\_Click(object sender, EventArgs e)

{

operation = 1;

size = 2;

Calculate(operation, size);

}

//при нажатии на кнопку произойдет присваивание размера и номера операции

private void разность\_Click(object sender, EventArgs e)

{

operation = 2;

size = 2;

Calculate(operation, size);

}

//при нажатии на кнопку произойдет присваивание размера и номера операции

private void Скаляр\_Click(object sender, EventArgs e)

{

operation = 3;

size = 2;

Calculate(operation, size);

}

//при нажатии на кнопку произойдет присваивание размера и номера операции

private void длиннаА\_Click(object sender, EventArgs e)

{

operation = 5;

size = 2;

Calculate(operation, size);

}

//при нажатии на кнопку произойдет присваивание размера и номера операции

private void длиннаB\_Click(object sender, EventArgs e)

{

operation = 6;

size = 2;

Calculate(operation, size);

}

//при нажатии на кнопку произойдет присваивание размера и номера операции

private void Сумма2\_Click(object sender, EventArgs e)

{

operation = 1;

size = 3;

Calculate(operation, size);

}

//при нажатии на кнопку произойдет присваивание размера и номера операции

private void Скаляр2\_Click(object sender, EventArgs e)

{

operation = 3;

size = 3;

Calculate(operation, size);

}

//при нажатии на кнопку произойдет присваивание размера и номера операции

private void Длинна\_а\_Click(object sender, EventArgs e)

{

operation = 5;

size = 3;

Calculate(operation, size);

}

//при нажатии на кнопку произойдет присваивание размера и номера операции

private void Длинна\_в\_Click(object sender, EventArgs e)

{

operation = 6;

size = 3;

Calculate(operation, size);

}

//при нажатии на кнопку произойдет присваивание размера и номера операции

private void Разн\_Click(object sender, EventArgs e)

{

operation = 2;

size = 3;

Calculate(operation, size);

}

//при нажатии на кнопку произойдет присваивание размера и номера операции

private void Векторн\_Click(object sender, EventArgs e)

{

operation = 4;

size = 3;

Calculate(operation, size);

}

//при нажатии на кнопку произойдет очищение всех ячеек

private void ОЧИСТИТЬ\_Click(object sender, EventArgs e)

{

operation = 0;

switch (typeA)

{

case false:

A1X.Text = "";

A2X.Text = "";

A1Y.Text = "";

A2Y.Text = "";

break;

case true:

AX.Text = "";

AY.Text = "";

break;

}

switch (typeB)

{

case false:

B1X.Text = "";

B2X.Text = "";

B1Y.Text = "";

B2Y.Text = "";

break;

case true:

BX.Text = "";

BY.Text = "";

break;

}

txtResult.Text = "";

}

//при нажатии на кнопку произойдет очищение всех ячеек

private void Очистить\_Click\_1(object sender, EventArgs e)

{

operation = 0;

switch (typeA)

{

case false:

A1\_X.Text = "";

A2\_X.Text = "";

A1\_Z.Text = "";

A1\_Y.Text = "";

A2\_Y.Text = "";

A2\_Z.Text = "";

break;

case true:

A\_X.Text = "";

A\_Y.Text = "";

A\_Z.Text = "";

break;

}

switch (typeB)

{

case false:

B1\_X.Text = "";

B2\_X.Text = "";

B1\_Z.Text = "";

B1\_Y.Text = "";

B2\_Y.Text = "";

B2\_Z.Text = "";

break;

case true:

B\_X.Text = "";

B\_Y.Text = "";

B\_Z.Text = "";

break;

}

txtResult2.Text = "";

}

//кнопка вызывающая форму "справка"

private void справкаToolStripMenuItem1\_Click(object sender, EventArgs e)

{

Form2 frm = new Form2();

frm.Show();

}

//кнопка заполняющая ячейки векторов рандомными числами

private void auto1\_Click(object sender, EventArgs e)

{

Random rand = new Random();

switch (typeA) {

case false:

//если вектор А задан 2мя точками то создаем вектор и записываем координаты рандомно

A1X.Text = "" + rand.Next(-10, 10);

A2X.Text = "" + rand.Next(-10, 10);

A1Y.Text = "" + rand.Next(-10, 10);

A2Y.Text = "" + rand.Next(-10, 10);

break;

case true:

//если вектор А задан координатами то создаем вектор и записываем координаты рандомно

AX.Text = "" + rand.Next(-10, 10);

AY.Text = "" + rand.Next(-10, 10);

break;

}

switch (typeB)

{

case false:

//если вектор В задан 2мя точками то создаем вектор и записываем координаты рандомно

B1X.Text = "" + rand.Next(-10, 10);

B2X.Text = "" + rand.Next(-10, 10);

B1Y.Text = "" + rand.Next(-10, 10);

B2Y.Text = "" + rand.Next(-10, 10);

break;

case true:

//если вектор В задан координатами то создаем вектор и записываем координаты рандомно

BX.Text = "" + rand.Next(-10, 10);

BY.Text = "" + rand.Next(-10, 10);

break;

}

}

//кнопка заполняющая ячейки векторов рандомными числами

private void auto3\_Click(object sender, EventArgs e)

{

Random rand = new Random();

switch (typeA)

{

case false:

//если вектор А задан 2мя точками то создаем вектор и записываем координаты рандомно

A1\_X.Text = "" + rand.Next(-10, 10);

A2\_X.Text = "" + rand.Next(-10, 10);

A1\_Y.Text = "" + rand.Next(-10, 10);

A2\_Y.Text = "" + rand.Next(-10, 10);

A1\_Z.Text = "" + rand.Next(-10, 10);

A2\_Z.Text = "" + rand.Next(-10, 10);

break;

case true:

//если векторА задан координатами то создаем вектор и записываем координаты рандомно

A\_X.Text = "" + rand.Next(-10, 10);

A\_Y.Text = "" + rand.Next(-10, 10);

A\_Z.Text = "" + rand.Next(-10, 10);

break;

}

switch (typeB)

{

case false:

//если вектор В задан 2мя точками то создаем вектор и записываем координаты рандомно

B1\_X.Text = "" + rand.Next(-10, 10);

B2\_X.Text = "" + rand.Next(-10, 10);

B1\_Y.Text = "" + rand.Next(-10, 10);

B2\_Y.Text = "" + rand.Next(-10, 10);

B1\_Z.Text = "" + rand.Next(-10, 10);

B2\_Z.Text = "" + rand.Next(-10, 10);

break;

case true:

//если вектор В задан то создаем вектор и записываем координаты рандомно

B\_X.Text = "" + rand.Next(-10, 10);

B\_Y.Text = "" + rand.Next(-10, 10);

B\_Z.Text = "" + rand.Next(-10, 10);

break;

}

}

//что бы сделать темную тему по щелчку,я меняю свойства background

private void темныйToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.BackColor = Color.FromArgb(77, 100, 141);

tabPage3.BackColor = Color.RoyalBlue;

tabPage4.BackColor = Color.RoyalBlue;

menuStrip1.BackColor = Color.FromArgb(77, 100, 141);

Acoord3.BackColor = Color.LightBlue;

Acoord.BackColor = Color.LightBlue;

Bcoord3.BackColor = Color.LightBlue;

Bcoord.BackColor = Color.LightBlue;

Apoint.BackColor = Color.LightBlue;

Apoint3.BackColor = Color.LightBlue;

Bpoint.BackColor = Color.LightBlue;

Bpoint3.BackColor = Color.LightBlue;

Скаляр2.BackColor = Color.LightBlue;

Сумма2.BackColor = Color.LightBlue;

сумма.BackColor = Color.LightBlue;

Разн.BackColor = Color.LightBlue;

разность.BackColor = Color.LightBlue;

Длинна\_а.BackColor = Color.LightBlue;

Длинна\_в.BackColor = Color.LightBlue;

длиннаB.BackColor = Color.LightBlue;

длиннаА.BackColor = Color.LightBlue;

Очистить.BackColor = Color.LightBlue;

Векторн.BackColor = Color.LightBlue;

auto1.BackColor = Color.LightBlue;

auto3.BackColor = Color.LightBlue;

Скаляр.BackColor = Color.LightBlue;

button1.BackColor = Color.LightBlue;

txtResult.BackColor = Color.LightBlue;

txtResult2.BackColor = Color.LightBlue;

}

//что бы сделать светлую тему по щелчку,я меняю свойства background

private void светлая\_Click(object sender, EventArgs e)

{

this.BackColor = Color.FromArgb(153, 103, 103);

tabPage3.BackColor = Color.FromArgb(215, 183, 163);

tabPage4.BackColor = Color.FromArgb(215, 183, 163);

menuStrip1.BackColor = Color.FromArgb(153, 103, 103);

Acoord3.BackColor = Color.FromArgb(234, 231, 234);

Acoord.BackColor = Color.FromArgb(234, 231, 234);

Bcoord3.BackColor = Color.FromArgb(234, 231, 234);

Bcoord.BackColor = Color.FromArgb(234, 231, 234);

Apoint.BackColor = Color.FromArgb(234, 231, 234);

Apoint3.BackColor = Color.FromArgb(234, 231, 234);

Bpoint.BackColor = Color.FromArgb(234, 231, 234);

Bpoint3.BackColor = Color.FromArgb(234, 231, 234);

Скаляр2.BackColor = Color.FromArgb(234, 231, 234);

Сумма2.BackColor = Color.FromArgb(234, 231, 234);

сумма.BackColor = Color.FromArgb(234, 231, 234);

Разн.BackColor = Color.FromArgb(234, 231, 234);

разность.BackColor = Color.FromArgb(234, 231, 234);

Длинна\_а.BackColor = Color.FromArgb(234, 231, 234);

Длинна\_в.BackColor = Color.FromArgb(234, 231, 234);

длиннаB.BackColor = Color.FromArgb(234, 231, 234);

длиннаА.BackColor = Color.FromArgb(234, 231, 234);

Очистить.BackColor = Color.FromArgb(234, 231, 234);

Векторн.BackColor = Color.FromArgb(234, 231, 234);

auto1.BackColor = Color.FromArgb(234, 231, 234);

auto3.BackColor = Color.FromArgb(234, 231, 234);

Скаляр.BackColor = Color.FromArgb(234, 231, 234);

button1.BackColor = Color.FromArgb(234, 231, 234);

txtResult.BackColor = Color.FromArgb(234, 231, 234);

txtResult2.BackColor = Color.FromArgb(234, 231, 234);

}

//комбобокс для выбора представления типа

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

switch (comboBox1.Text)

{

case "точкам":

Apoint.Visible = true;

Acoord.Visible = false;

typeA = false;

txtResult.Text = "";

break;

case "координатам":

Apoint.Visible = false;

Acoord.Visible = true;

typeA = true;

txtResult.Text = "";

break;

}

}

//комбобокс для выбора представления типа

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

switch (comboBox2.Text)

{

case "точкам":

Bcoord.Visible = false;

Bpoint.Visible = true;

typeB = false;

txtResult.Text = "";

break;

case "координатам":

Bpoint.Visible = false;// не видна

Bcoord.Visible = true;

typeB = true;

txtResult.Text = "";

break;

}

}

//комбобокс для выбора представления типа

private void comboBox3\_SelectedIndexChanged(object sender, EventArgs e)

{

switch (comboBox3.Text)

{

case "точкам":

Acoord3.Visible = false;

Apoint3.Visible = true;

typeA = false;

txtResult2.Text = "";

break;

case "координатам":

Apoint3.Visible = false;// не видна

Acoord3.Visible = true;

typeA = true;

txtResult2.Text = "";

break;

}

}

//комбобокс для выбора представления типа

private void comboBox4\_SelectedIndexChanged(object sender, EventArgs e)

{

switch (comboBox4.Text)

{

case "точкам":

Bcoord3.Visible = false;

Bpoint3.Visible = true;

typeB = false;

txtResult2.Text = "";

break;

case "координатам":

Bpoint3.Visible = false;// не видна

Bcoord3.Visible = true;

typeB = true;

txtResult2.Text = "";

break;

}

}

public void proverka(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if ((e.KeyChar <= 47 || e.KeyChar >= 58) && number != 8 && number != 44 && number != 45)

{

e.Handled = true;

}

}

private void onValueChanged(object sender, EventArgs e)

{

if(operation != 0) {

Calculate(operation,size);

}

}

}

}

**Код Form2.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace \_3\_4

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

Text = "Справка";

}

private void Form2\_Load(object sender, EventArgs e)

{

}

private void textBox1\_TextChanged\_1(object sender, EventArgs e)

{

}

}

}