Министерство транспорта Российской Федерации

Федеральное государственное автономное

образовательное учреждение высшего образования

«Российский университет транспорта (МИИТ)»

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа №3

по дисциплине:

«Методы программирования»

на тему:  
«Автоматизация компоновки ГПИ. Функции»

Выполнил: ст. гр. ТКИ-341

Панаргин В.М.

Вариант №5

Проверил: к.т.н., доцент Сафронов А.И.

Москва – 2024 г.

# **1. Цель работы**

Закрепить навыки разработки визуального пользовательского интерфейса, освоить работу с текстовыми файлами в среде *Microsoft Visual Studio*, научиться взаимно увязывать одни элементы управления с другими, получить представление о перерисовке и перемасштабировании».

# **2. Формулировка задачи**

«В интегрированной среде разработки *Visual Studio* разработать программу в режиме *Windows Forms Application* на языке *Visual C#*, представляющую собой пользовательский интерфейс, содержащий главное меню, позволяющее:

1. Начать работу с приложением.
2. Прервать работу приложения.
3. Предоставить пользователю справочную информацию о работе с приложением.

Сама программа должна реализовывать вывод в списки значений аргумента и соответствующих им значений функций. Список функций должен обязательно содержать следующие пункты:

1. Логарифм по основанию 2,
2. Тангенс,
3. Возведение в квадрат,
4. Косинус,
5. Натуральный логарифм,
6. Арккосинус,
7. Извлечение корня,
8. Арктангенс,
9. Синус,
10. Десятичный логарифм,
11. Арксинус.

Согласно заданию, список должен быть организован в виде обыкновенного списка. Функции из списка должны быть расположены в следующем порядке: 7, 3, 8, 4, 6, 2, 11, 9, 10, 5, 1. Так же необходимо поместить дополнительную функциональную зависимость 1 / cos(x) (секанс) на 2 позицию. Таким образом, функции должны быть расположены в следующем порядке: Извлечение корня, Секанс, Возведение в квадрат, Арктангенс, Косинус, Арккосинус, Тангенс, Арксинус, Синус, Натуральный логарифм, Логарифм по основанию 2.

# **3. Составление диаграммы классов, входящих в состав решения.**

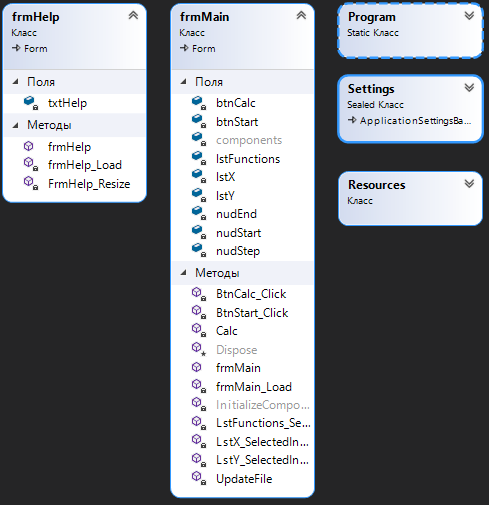


Рисунок 1 – Диаграмма классов

# **4. Составление сети Петри запрограммированного технологического процесса.**

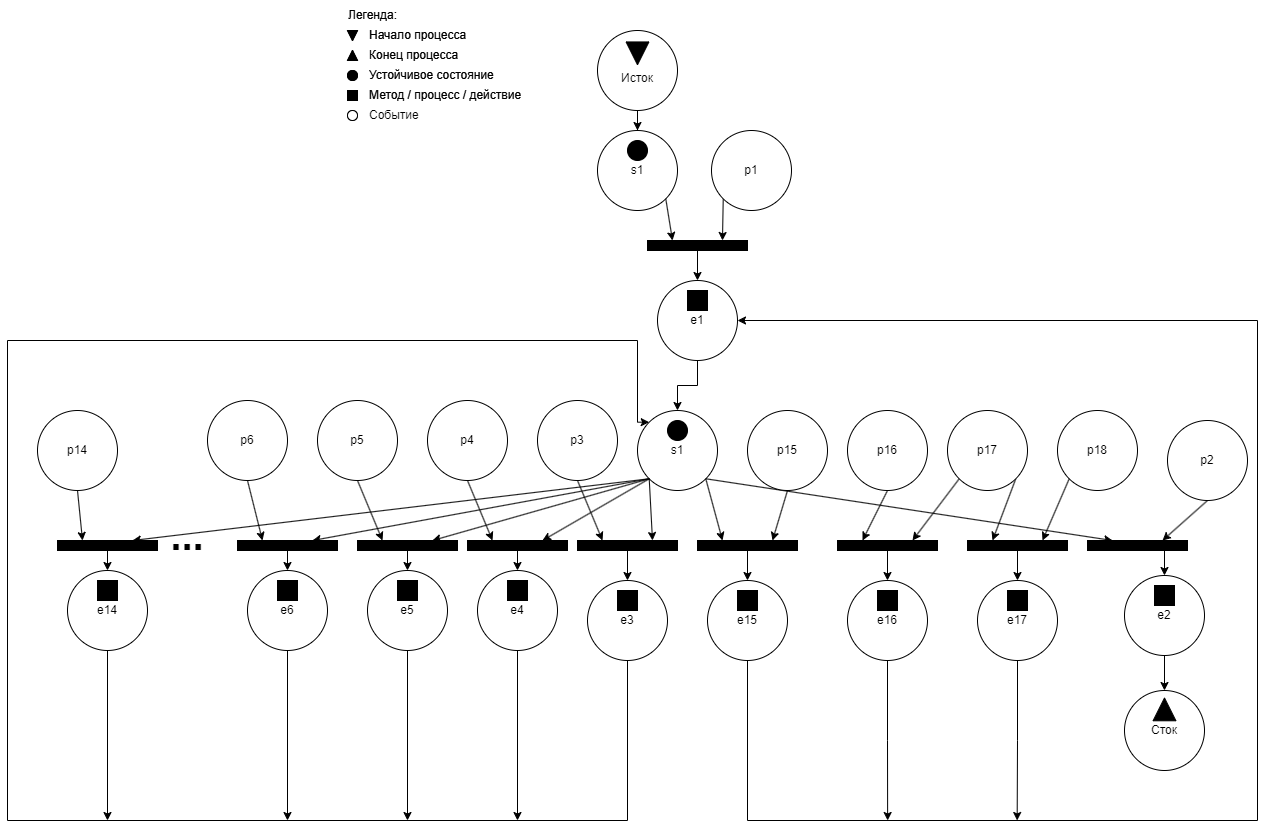


Рисунок 2 - Сеть Петри основной формы

**Описание сети Петри**

- состояния (states)

s1 – форма ожидает действий пользователя

- действия (effects)

e1 – разблокируются кнопки взаимодействия;

кнопка «Начало работы с приложением» деактивируется

e2 – закрытие формы

e3 – запись в файл аргументов и результатов функции извлечение корня

e4 – запись в файл аргументов и результатов функции секанс

e5 – запись в файл аргументов и результатов функции возведение в квадрат

e6 – запись в файл аргументов и результатов функции арктангенс

…

e14 – запись в файл аргументов и результатов функции логарифм по основанию 2

e15 – вывод результатов из файла в списки

e16 – выбор соответствующего элемента из списка ординат

e17 – выбор соответствующего элемента из списка абцисс

- события (prompts)

p1 – нажата кнопка «Начало работы с приложением»

p2 – нажата кнопка закрытия

p3 – в списке выбрана функция извлечение корня

p4 – в списке выбрана функция секанс

p5 – в списке выбрана функция возведение в квадрат

p6 – в списке выбрана функция арктангенс

…

p14 – в списке выбрана функция логарифм по основанию 2

p15 – нажатие кнопки рассчитать

p16 – пользователь выбрал элемент из списка абцисс

p17 – списки заполнены значениями

p18 – пользователь выбрал элемент из списка ординат

# **5. Составление схем алгоритмов методов в составе решения, отмеченных на сети Петри в качестве «эффектов» (метка ).**



Рисунок 3 – Алгоритм кнопки "Начало работы с приложением"

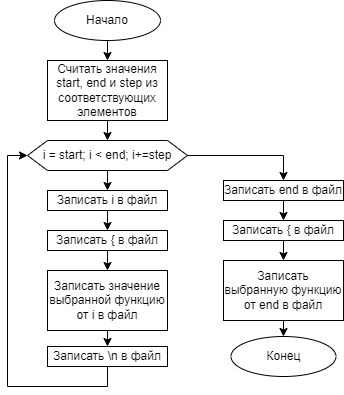


Рисунок 4 - Алгоритм записи информации в файл



Рисунок 5 - Алгоритм чтения информации из файла

# **6. Подбор тестовых примеров.**

Тестовые примеры:

* Проверить корректность вычисления каждой из формул
* Проверить запись в файл

**7. Листинг (код) составленного программного обеспечения.**

Основная форма:

public partial class frmMain : Form

{

private ListBox lstFunctions;

private ListBox lstX;

private ListBox lstY;

private NumericUpDown nudStart;

private NumericUpDown nudEnd;

private NumericUpDown nudStep;

private Button btnCalc;

private Button btnStart;

public frmMain()

{

InitializeComponent();

}

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

{

this.Text = "Задание №3 выполнил: Панаргин Владислав Максимович; Номер варианта: 5; Дата выполнения: 17/03/2024";

this.Cursor = Cursors.Hand;

this.BackColor = SystemColors.ControlText;

this.ClientSize = new Size(400, 310);

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.lstFunctions = new ListBox();

this.lstX = new ListBox();

this.lstY = new ListBox();

this.nudStart = new NumericUpDown();

this.nudEnd = new NumericUpDown();

this.nudStep = new NumericUpDown();

this.btnCalc = new Button();

this.btnStart = new Button();

//lstFunctions

this.lstFunctions.Location = new System.Drawing.Point(10,10);

this.lstFunctions.Size = new System.Drawing.Size(160,170);

this.lstFunctions.Items.AddRange(new string[] { "Извлечение корня", "Секанс", "Возведение в квадрат", "Арктангенс", "Косинус", "Арккосинус", "Тангенс", "Арксинус", "Синус", "Десятичный логарифм", "Натуральный логарифм", "Логарифм по основанию 2" });

this.lstFunctions.SelectedIndexChanged += LstFunctions\_SelectedIndexChanged;

this.lstFunctions.SelectedIndex = 0;

//lstX

this.lstX.Location = new Point(180, 10);

this.lstX.Size = new Size(100, 300);

this.lstX.SelectedIndexChanged += LstX\_SelectedIndexChanged;

//lstY

this.lstY.Location = new Point(290, 10);

this.lstY.Size = new Size(100, 300);

this.lstY.SelectedIndexChanged += LstY\_SelectedIndexChanged;

//nudStart

this.nudStart.Location = new Point(10, 190);

this.nudStart.Size = new Size(50, 20);

this.nudStart.Minimum = -100;

this.nudStart.Maximum = 100;

this.nudStart.Value = -5;

this.nudStart.DecimalPlaces = 1;

this.nudStart.Increment = (decimal)0.1;

//nudEnd

this.nudEnd.Location = new Point(65, 190);

this.nudEnd.Size = new Size(50, 20);

this.nudEnd.Minimum = -100;

this.nudEnd.Maximum = 100;

this.nudEnd.Value = 5;

this.nudEnd.DecimalPlaces = 1;

this.nudEnd.Increment = (decimal)0.1;

//nudStep

this.nudStep.Location = new Point(120, 190);

this.nudStep.Size = new Size(50, 20);

this.nudStep.Minimum = (decimal)0.1 ;

this.nudStep.Maximum = 10;

this.nudStep.Value = (decimal)0.1;

this.nudStep.DecimalPlaces = 1;

this.nudStep.Increment = (decimal)0.1;

//btnCalc

this.btnCalc.Location = new Point(10, 220);

this.btnCalc.Size = new Size(160, 80);

this.btnCalc.FlatStyle = FlatStyle.Flat;

this.btnCalc.FlatAppearance.BorderSize = 1;

this.btnCalc.Text = "Рассчитать";

this.btnCalc.Click += BtnCalc\_Click;

//btnStart

this.btnStart.Location = new Point(100, 135);

this.btnStart.Size = new Size(200, 40);

this.btnStart.FlatStyle = FlatStyle.Flat;

this.btnStart.FlatAppearance.BorderSize = 1;

this.btnStart.Text = "Начало работы с приложением";

this.btnStart.Click += BtnStart\_Click;

this.Controls.AddRange(new Control[] { lstFunctions, lstX, lstY, nudStart, nudEnd, nudStep, btnCalc });

foreach (Control control in this.Controls)

{

control.Visible = false;

}

this.Controls.Add(btnStart);

foreach (Control control in this.Controls)

{

control.BackColor = SystemColors.ControlText;

control.ForeColor = SystemColors.Window;

}

frmHelp fh = new frmHelp();

fh.Owner = this;

fh.ShowDialog();

fh.Dispose();

}

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

{

foreach (Control control in this.Controls)

{

control.Visible = true;

}

this.btnStart.Visible = false;

}

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

{

this.lstX.SelectedIndex = this.lstY.SelectedIndex;

}

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

{

this.lstY.SelectedIndex = this.lstX.SelectedIndex;

}

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

{

this.UpdateFile();

this.lstX.Items.Clear();

this.lstY.Items.Clear();

StreamReader sr = new StreamReader("E:\\Методы программирования\\Задание 3\\Task3\\FuncArgs.txt");

while (true)

{

string str = sr.ReadLine();

if (str == null) break;

string[] strarr = str.Split('{');

this.lstX.Items.Add(strarr[0]);

this.lstY.Items.Add(strarr[1]);

}

sr.Close();

}

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

{

this.UpdateFile();

}

private void UpdateFile()

{

StreamWriter sw = new StreamWriter("E:\\Методы программирования\\Задание 3\\Task3\\FuncArgs.txt");

decimal start = this.nudStart.Value;

decimal end = this.nudEnd.Value;

decimal step = this.nudStep.Value;

for (decimal i = start; i < end; i+=step)

{

sw.Write(i);

sw.Write("{");

sw.Write(this.Calc(i));

sw.Write("\n");

}

sw.Write(end);

sw.Write("{");

sw.Write(this.Calc(end));

sw.Close();

}

private string Calc(decimal arg)

{

switch (this.lstFunctions.SelectedIndex)

{

case 0:

if (arg >= 0) return Math.Sqrt((double)arg).ToString("F1");

else return "UNDEFINED";

case 1:

if (Math.Cos((double)arg) != 0) return (1 / Math.Cos((double)arg)).ToString("F1");

else return "UNDEFINED";

case 2:

return Math.Pow((double)arg, 2).ToString("F1");

case 3:

return Math.Atan((double)arg).ToString("F1");

case 4:

return Math.Cos((double)arg).ToString("F1");

case 5:

if (Math.Abs(arg) > 1) return "UNDEFINED";

else return Math.Acos((double)arg).ToString("F1");

case 6:

if (Math.Cos((double)arg) != 0) return Math.Tan((double)arg).ToString("F1");

else return "UNDEFINED";

case 7:

if (Math.Abs(arg) > 1) return "UNDEFINED";

else return Math.Asin((double)arg).ToString("F1");

case 8:

return Math.Sin((double)arg).ToString("F1");

case 9:

if (arg > 0) return Math.Log((double)arg, 10).ToString("F1");

else return "UNDEFINED";

case 10:

if (arg > 0) return Math.Log((double)arg, Math.E).ToString("F1");

else return "UNDEFINED";

default:

if (arg > 0) return Math.Log((double)arg, 2).ToString("F1");

else return "UNDEFINED";

}

}

}

Дополнительная форма:

public partial class frmHelp : Form

{

private TextBox txtHelp;

public frmHelp()

{

this.txtHelp = new TextBox();

this.SuspendLayout();

//

// txtHelp

//

this.txtHelp.Location = new System.Drawing.Point(0, 0);

this.txtHelp.Size = new System.Drawing.Size(800, 450);

this.txtHelp.Multiline = true;

this.txtHelp.ReadOnly = true;

//

// frmHelp

//

this.ClientSize = new System.Drawing.Size(800, 100);

this.Controls.Add(this.txtHelp);

this.Name = "frmHelp";

this.Text = "Окно со справочной информацией";

this.Cursor = Cursors.Hand;

this.Load += new System.EventHandler(this.frmHelp\_Load);

this.ResumeLayout(false);

}

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

{

StreamReader sr = new StreamReader("E:\\Методы программирования\\Задание 3\\Task3\\HelpInformation.txt");

txtHelp.Text = sr.ReadToEnd();

sr.Close();

this.Resize += FrmHelp\_Resize;

}

private void FrmHelp\_Resize(object sender, EventArgs e)

{

this.txtHelp.Size = this.ClientSize;

}

}

# **8. Графический пользовательский интерфейс программного обеспечения и его описание.**

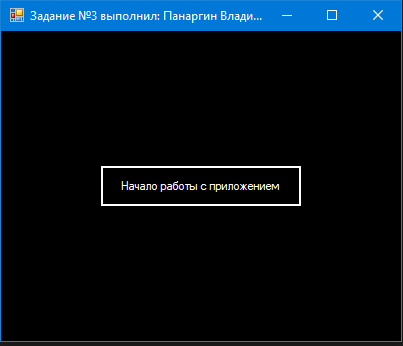


Рисунок 6 - Интерфейс основной формы до нажатия кнопки «Начало работы с приложением»

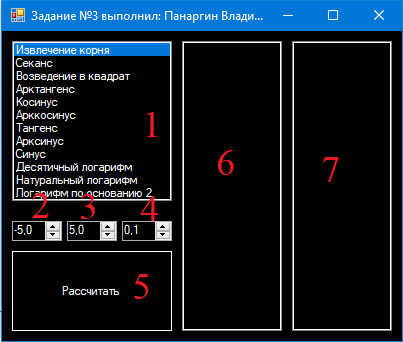


Рисунок 7 – Интерфейс основной формы после нажатия кнопки «Начало работы с приложением

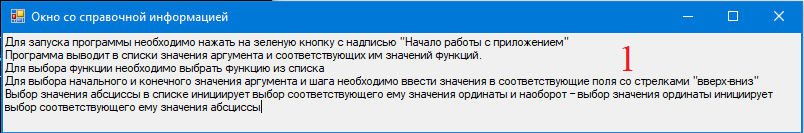
1. Список функций
2. Поле для выбора параметра start
3. Поле для выбора параметра end
4. Поле для выбора параметра step
5. Кнопка расчета
6. Список абсцисс
7. Список ординат

Рисунок 8 – Интерфейс вспомогательной формы

1. Зона для выгрузки текста из файла

# **9. Подтверждение соответствия графического пользовательского интерфейса требованиям к оформлению.**

1. Заголовок экранной формы должен содержать надпись вида: «Задание №3 выполнил: [Фамилия И.О. автора]; Номер варианта: [Номер]; Дата выполнения: [дд/мм/гггг].



Рисунок 9 - Название формы

2. Дата выполнения проставляется в момент, когда программа считается законченной и по ней можно готовить итоговый отчёт о выполнении работы.



Рисунок 10 - Дата на момент написания отчета

4. Нечётные варианты заменяют стандартный курсор экранной формы со «стрелки» на «руку».



Рисунок 11 - Фрагмент кода с изменением вида курсора

5. Справочная информация должна быть вызвана в дочерней экранной форме и считана в статическое по размеру текстовое поле с выставленным запретом на редактирование текстовой информации (нечётные варианты).

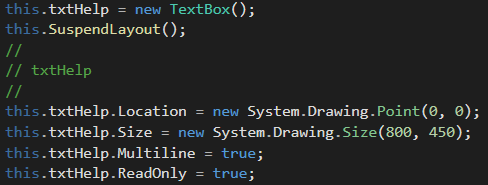


Рисунок 12 - Фрагмент кода с созданием текстового поля для выгрузки справочной информации

6. В режиме конструктора экранная форма не должна содержать ни одного элемента управления.

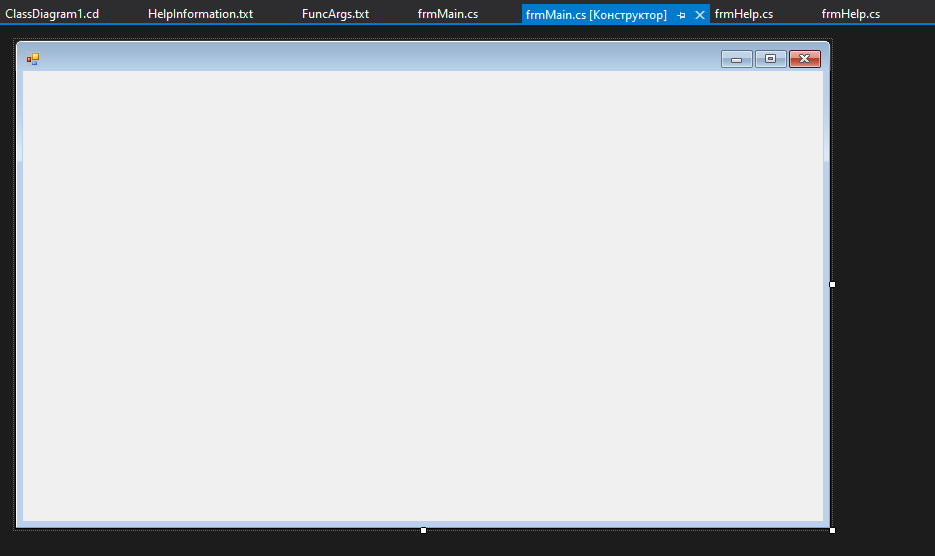


Рисунок 13 - Конструктор главной формы

7. Решение (*solution*) должно содержать только одну единственную экранную форму. Экранная форма со справкой настраивается программным кодом.

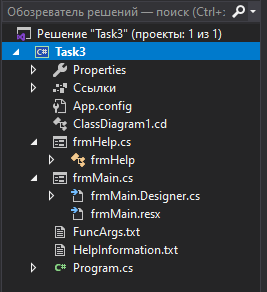


Рисунок 14 - Решение (solution)

8. Компоненты экранной формы со справкой должны быть чувствительными к изменению размера экранной формы по ширине и высоте (обязательно подключение события *Resize*).

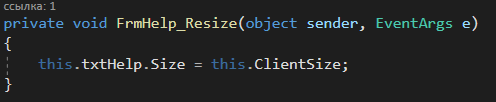


Рисунок 15 - Фрагмент кода с изменением размера компонента экранной формы со справкой

9. Названия функций должны быть элементами обыкновенного списка (нечётные варианты).

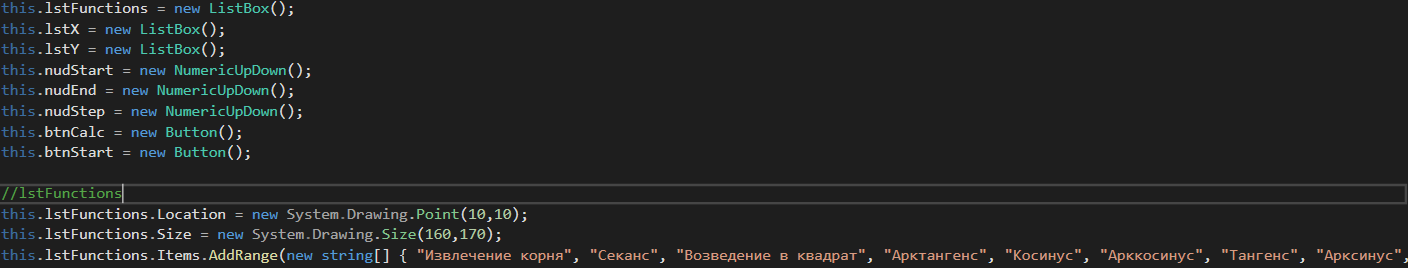


Рисунок 16 - Фрагмент кода с созданием списка функций

10. Выбор конкретной функции из списка инициирует заполнение некоторого файла значениями аргумента и значениями функции. Разделитель данных подбирается согласно таблице вариантов.

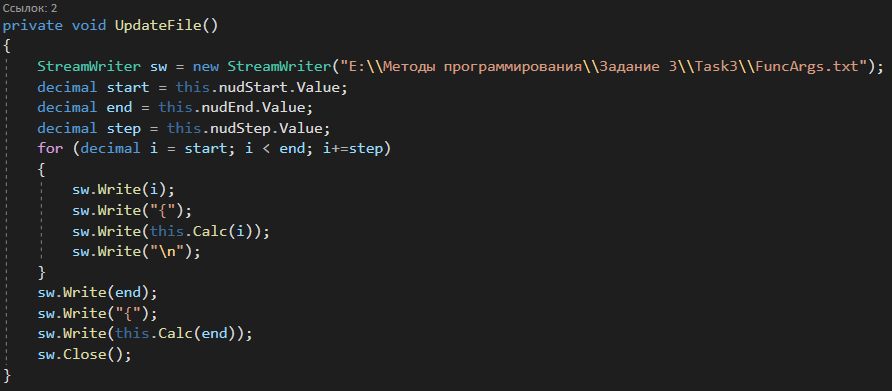


Рисунок 17 - Фрагмент кода с функцией заполнения файла

Таблица 2 – Варианты разделителей, записываемых в файл

|  |  |
| --- | --- |
| 1. | $ |
| 2. | % |
| 3. | &\* |
| 4. | <> |
| 5. | { |
| 6. | $#@ |
| 7. | ^^ |
| 8. | ( |
| 9. | . |
| 10. | >> |
| 11. | , |
| 12. | ; |
| 13. | : |
| 14. | ++ |
| 15. | = |
| 16. | !@ |
| 17. | ||| |
| 18. | пробел |
| 19. | \_ |
| 20. | … |

11. Нечётные варианты заменяют цвет подложки экранной формы в соответствии с представленной ниже таблицей.



Рисунок 18 - Фрагмент кода с выбором цвета подложки

Таблица 3 – Варианты цветового оформления подложки

|  |  |
| --- | --- |
| 1. | *ActiveCaption* |
| 3. | *AppWorkspace* |
| 5. | *ControlText* |
| 7. | *Desktop* |
| 9. | *GradientActiveCaption* |
| 11. | *Highlight* |
| 13. | *HotTrack* |
| 15. | *Info* |
| 17. | *MenuHighlight* |
| 19. | *Window* |

Подобрать читаемый цвет текста к предложенной по варианту схеме.

12. Предусмотреть кнопку, нажатие на которую инициирует заполнение двух списков (один список под значения аргумента, другой список – под значения функции, округлённых по правилу *(N mod 5) + 1* для знаков после плавающей запятой, где *N* – номер варианта) данными о функциональной зависимости из файла. Обучающиеся с чётными номерами вариантов обновляют существующие основные списки, с нечётными номерами вариантов выводят значения, считанные из файла, в дополнительные списки.

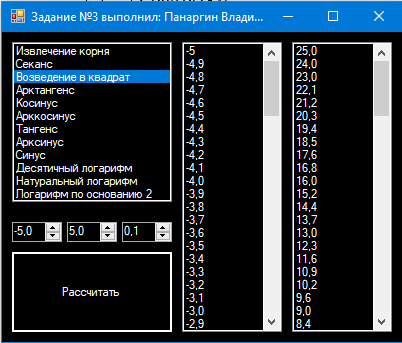


Рисунок 19 - Результат работы функции расчета

13. В качестве исходного состояния функциональных компонентов главной экранной формы принимается невидимое состояние (*.Visible = false*) – нечётные варианты.

см. Рисунок 6

14. Кнопка «Начало работы с приложением» должна реализовывать активацию доступа пользователя к функциональным компонентам или отображение функциональных компонентов на экрвнной форме для пользователя.

см. Рисунок 7

15. Все элементы программы должны носить значащие имена переменных, в которых отражено существо этих элементов, например, экранная форма – *frmMain*, ярлык – *lblHelp*, комбинированный список – *cmbFigures* и т.д.

см. Диаграмма классов

16. Выбор значения абсциссы в списке должен инициировать выбор соответствующего ему значения ординаты и наоборот – выбор значения ординаты должен инициировать выбор соответствующего ему значения абсциссы (работа с параметрами *SelectedIndex* или *SelectedItem* списков).

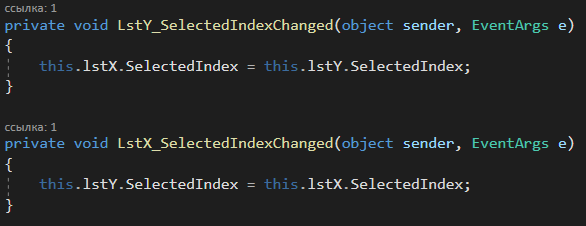


Рисунок 20 - Фрагмент кода с функциями обработки выбора значения в списках абцисс или ординат

# **10. Расчёт тестовых примеров с использованием составленного программного обеспечения.**

* Проверить корректность вычисления каждой из формул

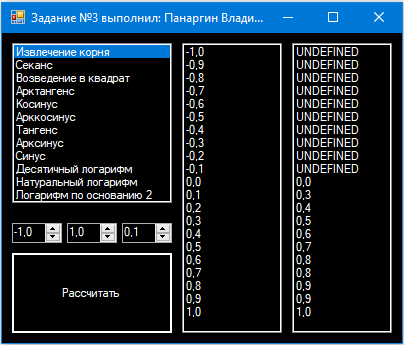


Рисунок 21 - Вычисление функции извлечение корня

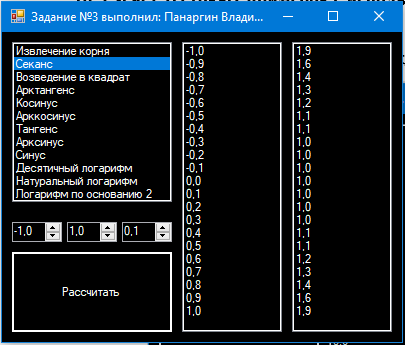


Рисунок 22 - Вычисление функции секанс

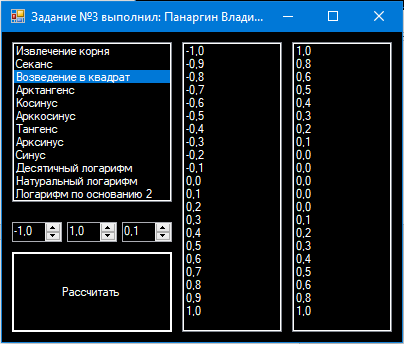


Рисунок 23 - Вычисление функции возведение в квадрат

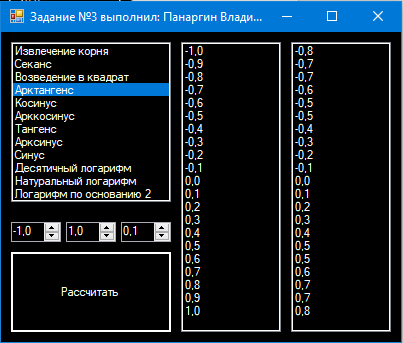


Рисунок 24 - Вычисление функции арктангенс

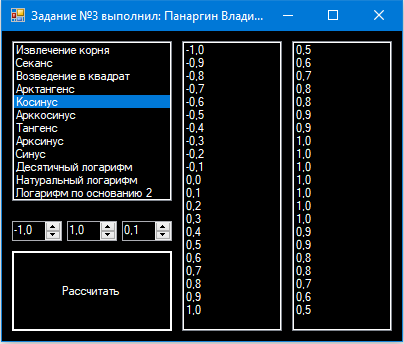


Рисунок 25 - Вычисление функции косинус

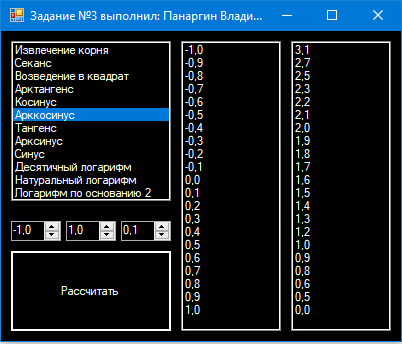


Рисунок 26 - Вычисление функции арккосинус

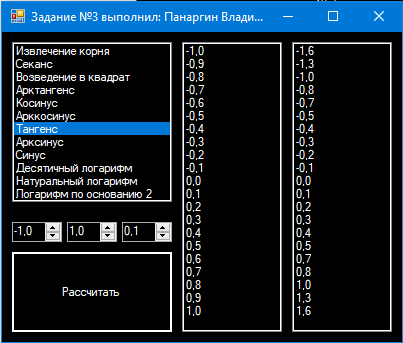


Рисунок 27 - Вычисление функции тангенс

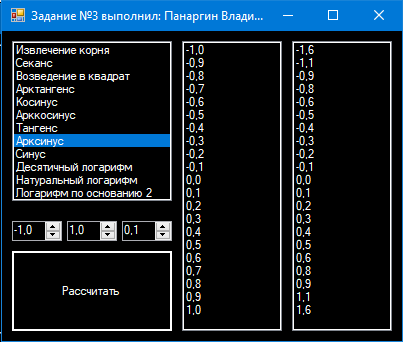


Рисунок 28 - Вычисление функции арксинус

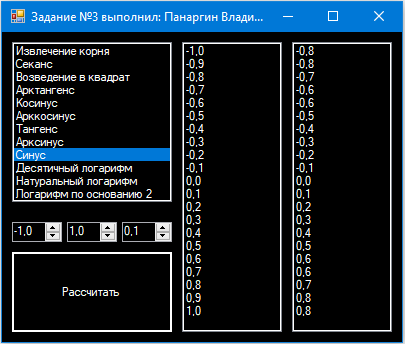


Рисунок 29 - Вычисление функции синус

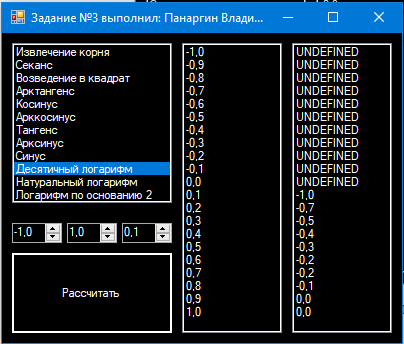


Рисунок 30 - Вычисление функции десятичный логарифм

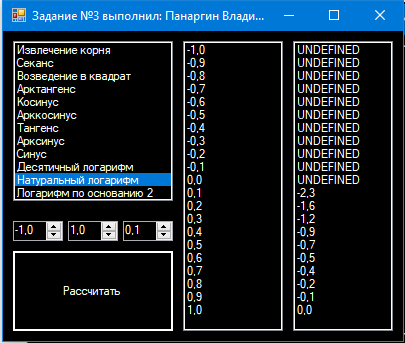


Рисунок 31 - Вычисление функции натуральный логарифм

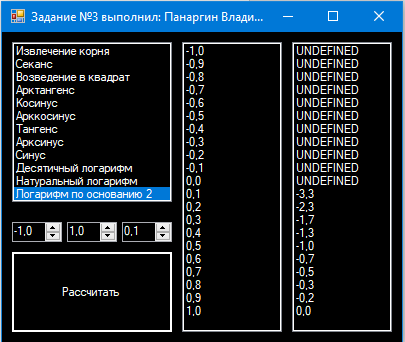


Рисунок 32 - Вычисление функции логарифм по основанию 2

* Проверить запись в файл

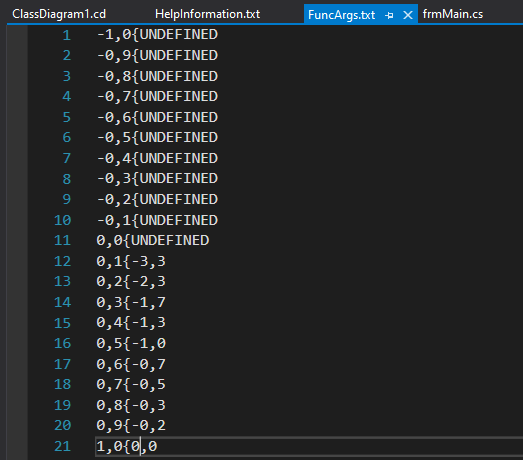


Рисунок 33 - Записи в файле

# **11. Формулировка вывода о проделанной работе**

В рамках данной работы были закреплены навыки разработки визуального пользовательского интерфейса, освоена работа с текстовыми файлами и кодировкой в среде *Microsoft* *Visual Studio*, были получены знания о взаимном увязывании одних элементов управления с другими, представление о перерисовке и перемасштабировании.