# 2 РАБОЧИЙ ПРОЕКТ

## 2.1 Общие сведения о работе программы

В результате выполнения курсовой работы была создана программа, позволяющая проводить автоматические тестирования на ввод логических. В программе реализованы алгоритмы генерации логических выражений, генерация необходимой области для логического выражения, сравнения введенного пользователем и сгенерированного выражения. Программа разработана на языке программирования высокого уровня *C#* с использованием средств .NET Framework версии 3,5 и системы Windows Presentation Foundation (WPF) в интегрированной среде *Microsoft Visual Studio 2013* и работает под управлением операционных систем семейства Windows.

* 1. **Функциональное назначение**

Программа обеспечивает следующие функциональные возможности:

* проведение тестирования на ввод логических выражений;
* проведение тестирования;
* вывод результатов тестирования.

**Функциональные ограничения**

Количество заданий в базе не должно быть меньше 10.

* 1. **Инсталляция и выполнение**

Для установки программного продукта необходимо запустить файл LogicSetup.exe и следовать подсказкам системы. После установки на рабочем столе появится ярлык «Проверка знания булевых выражений» (если было выбрано создание ярлыка) либо в установочной папке запустить файл «Проверка знания булевых выражений».

* 1. **Общий алгоритм работы программного продукта**

Блок-схема, демонстрирующая общий алгоритм работы программы, представлена в приложении на рисунке П.1.

В программе реализованы следующие модули:

1. MainWindow.xaml.
2. MainWindow.xaml.cs.
3. Test.xaml.
4. Test.xaml.cs.
5. Result.xaml.
6. Result.xaml.cs.
7. RPN.cs.
8. TestChec.cs.
9. from\_XML\_to\_Task.cs.
10. Task.cs.
11. Primitive.cs.
12. polyline.cs.
13. poligon.cs.
14. line.cs.
15. ellips.cs.
16. lable.cs
17. point.cs

Функциональное назначение модулей представлено в таблицах 2.1-2.12.

Модуль MainWindow*.xaml.cs* содержит класс MainWindow, который реализует логику работы модуля MainWindow*.xaml,* описывающего процесс навигации в главном меню.

Модуль *Test.xaml.cs* содержит класс *Test*, который реализует логику работы модуля *Test.xaml,* описывающего процесс прохождения тестирования.

В таблице 2.2 приведены методы класса *Test*.

Таблица 2.2 – Описание класса *Test*

|  |  |
| --- | --- |
| **Прототип** | **Назначение** |
| **public void drawXOY(Canvas draw)** | Рисует оси координат. |

Модуль Result*.cs* содержит класс Result, которыйреализует вывод результата тестирования на экран с последующим сохранением его в файл статистики.

В таблице 2.3 приведены методы класса Result.

Таблица 2.3 – Описание методов класса Result

|  |  |
| --- | --- |
| **Прототип** | **Назначение** |
| **private void addStatistic()** | Добавляет статистику тестирования в файл статистики. |

Модуль RPN*.cs* содержит класс RPN, которыйреализует перевод логического выражения из инфиксной записи в постфиксную.

В таблице 2.4 приведены методы класса RPN.

Таблица 2.4 – Описание методов класса RPN

|  |  |
| --- | --- |
| **Прототип** | **Назначение** |
| **private bool isANumber(string value)** | Проверяет символ на принадлежность к целым числам. |
| **private string retEndDel(string returnString)** | Возвращает последний символ строки и удаляет его. |
| **public string inPostfix(string IN)** | Переводит строку в постфиксную запись. |

Модуль *Task.cs* содержит класс *Task*, которыйпредставляет из себя задание для тестирования.

В таблице 2.5 приведены методы класса *Task*.

Таблица 2.5 – Описание методов класса *Task*

|  |  |
| --- | --- |
| **Прототип** | **Назначение** |
| **public void Draw(Canvas myCanvas)** | Выводит на экран задания. |

Модуль from\_XML\_to\_Task*.cs* содержит класс from\_XML\_to\_Task, которыйреализует разбор базы шаблонов на задания.

В таблице 2.6 приведены методы класса from\_XML\_to\_Task.

Таблица 2.6 – Описание методов класса from\_XML\_to\_Task

|  |  |
| --- | --- |
| **Прототип** | **Назначение** |
| **public List<Task> Decode()** | Декодирование файла с базой шаблонов. |
| **private bool isInCircle(int x, int y, int x0, int y0, int r)** | Проверка на принадлежность точки кругу. |

Модуль TestChec*.cs* содержит класс TestChec, которыйреализует проверку введенных пользователем ответов.

В таблице 2.7 приведены методы класса TestChec.

Таблица 2.7 – Описание методов класса TestChec

|  |  |
| --- | --- |
| **Прототип** | **Назначение** |
| **public bool getChek(string Etal, string User)** | Сверяет строку эталонного решения с ответом пользователя. |

­­­­­­­­­­­­­­

Модуль *MathExprTestingState.cs* содержит классы MathExprTestingState и MathExprTestStepDetails, которые являются структурами для хранения данных о текущем состоянии тестирования и о выполняемом в данный момент времени задании. Данные классы не содержат методов.

Модуль *MathTestResultsBigTable.cs* содержит класс MathTestResultsBigTable, которыйреализует возможность вывода на экран таблицы с подробной информацией о каждом задании проведенного тестирования.

В таблице 2.8 приведены методы класса MathException.

Таблица 2.8 – Описание методов класса MathException

|  |  |
| --- | --- |
| **Прототип** | **Назначение** |
| **MathTestResultsBigTable (List<MathExprTestDetails> dtls)** | Конструктор класса, инициализирующий таблицу |
| **void AddHeader(Grid table)** | Добавляет первую строку с названиями колонок |
| **void AddRow(Grid table, MathExprTestDetails dtls, int ctr)** | Добавляет строку со значениями в таблицу |
| **List<string> StylishFunctions (List<string> expression)** | Преобразует системные идентификаторы функций в их понятные названия |

Модуль *MathTestResultStars.cs* содержит класс MathTestResultStars, которыйреализует возможность вывода на экран оценки в графическом виде (в виде звездочек).

В таблице 2.9 приведены методы класса MathTestResultStars.

Таблица 2.9 – Описание методов класса MathTestResultStars

|  |  |
| --- | --- |
| **Прототип** | **Назначение** |
| **MathTestResultStars(int stars)** | Инициализирует данный элемент интерфейса |
| **Image GetStar(Uri uri)** | Возвращает рисунок со звездочкой |
| **Image BronzeStar()** | Возвращает бронзовую звездочку |
| **Image SilverStar()** | Возвращает серебряную звездочку |
| **Image GoldStar()** | Возвращает золотую звездочку |

Модуль *MathTestStepsTable.cs* содержит класс MathTestStepsTable, которыйреализует возможность вывода на экран текущего состояния тестирования.

В таблице 2.10 приведены методы класса MathTestStepsTable.

Таблица 2.10 – Описание методов класса MathTestStepsTable

|  |  |
| --- | --- |
| **Прототип** | **Назначение** |
| **MathTestStepsTable(int steps)** | Инициализирует данный элемент интерфейса |
| **void DrawStepsBlocks()** | Выводит на экран блоки в количестве, равном количеству заданий в тестировании |
| **void LightUpCurrentStep()** | Подсвечивает на экране блок с номером текущего задания |
| **void NextStep(bool correct, MathExceptionReasons falutReason, string failedToken)** | Обрабатывает переход к следующему заданию |

Модуль *Spinner.cs* содержит абстрактный класс Spinner и его классы-наследники PercentSpinner, FloatSpinner и IntSpinner, которыереализуют возможность вывода на экран элементов управления с содержимым различного типа для настройки параметров тестирования.

В таблице 2.11 приведены методы абстрактного класса Spinner.

Таблица 2.11 – Описание методов абстрактного класса Spinner

|  |  |
| --- | --- |
| **Прототип** | **Назначение** |
| **bool CheckIfValid()** | Проверяет, содержит ли данный элемент управления допустимое значение. |
| **string GetValue()** | Возвращает содержимое элемента управления |
| **void IndicateError()** | Преобразует внешний вид элемента, сигнализируя пользователя о некорректном значении. |
| **void IndicateNormal()** | Преобразует внешний вид элемента в нормальный вид, сигнализируя о корректном значении |

Также классы PercentSpinner, FloatSpinner и IntSpinner содержат дополнительно свои методы, приведенные в таблице 2.12.

Таблица 2.12 – Описание методов классов-наследников класса Spinner

|  |  |
| --- | --- |
| **Прототип** | **Назначение** |
| **PercentSpinner(string lbl, float vl)**  **FloatSpinner(string lbl, float val)**  **IntSpinner(string lbl, int val)** | Конструкторы данных классов |
| **void SetValue(float value)**  **void SetValue(float value)**  **void SetValue(int value)** | Задает значение данного элемента управления |
| **void OnValueChanged()** | Обработчик события изменения содержимого данного элемента |

* 1. **Разработанные меню и интерфейсы**

Для запуска программы необходимо запустить среду Semantic IDE.

Работа в программе возможна в одном из двух режимов – режим прохождения тестирования и режим настройки параметров тестирования и генерации выражений.

Основное меню программы находится на основной ленте интерфейса Semantic IDE.

Главное окно среды и меню тестирования на ленте показаны на рисунке 2.2.

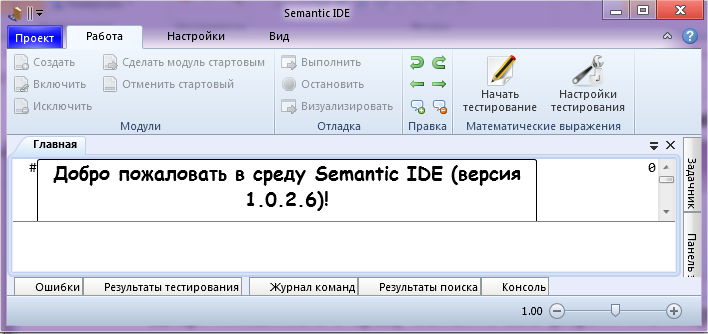


Рисунок 2.2 – Основное окно Semantic IDE и меню программы на основной ленте

Основное меню программы содержит следующие кнопки:

1. кнопка перехода в режим тестирования;
2. кнопка настроек тестирования.

Чтобы начать тестирование, необходимо нажать кнопку «Начать тестирование». Чтобы перейти в настройки тестирования, необходимо нажать кнопку «Настройки тестирования».

* + 1. **Интерфейс окна режима тестирования**

После нажатия кнопки «Начать тестирование» пользователь попадает в окно режима тестирования, которое продемонстрированно на рисунке 2.3.

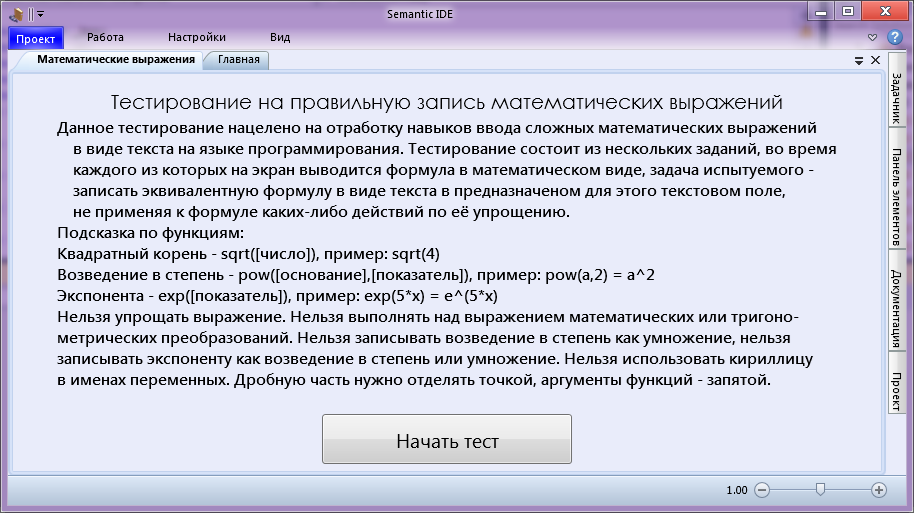


Рисунок 2.3 – Интерфейс окна режима тестирования

Окно режима тестирования содержит текст описания тестирования, подсказку по вводу функций и кнопку начала теста.

После нажатия на кнопку начала теста пользователь попадает в режим прохождения тестирования, показанный на рисунке 2.4.

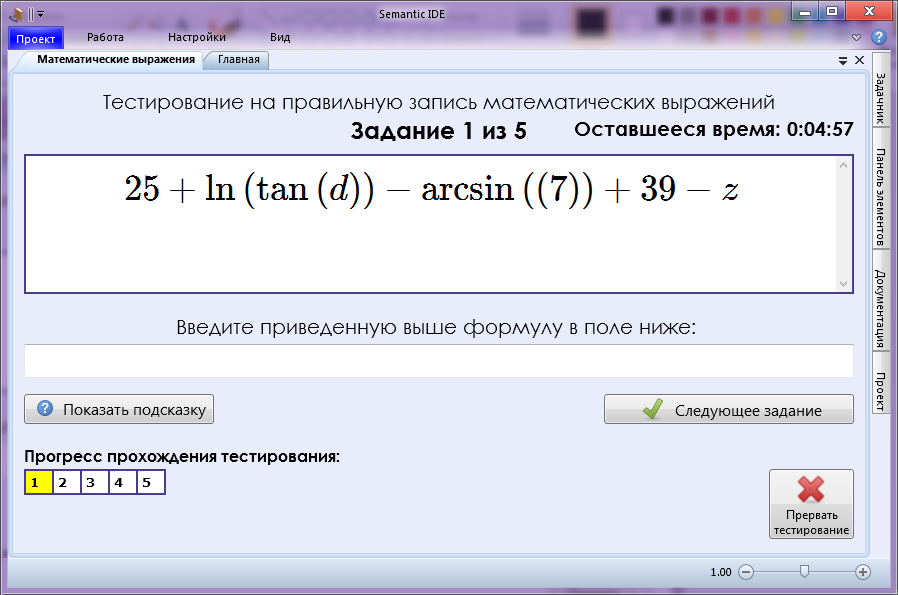


Рисунок 2.4 – Окно режима прохождения тестирования

Окно режима прохождения тестирования содержит следующие элементы:

1. Номер текущего задания и общее количество заданий;
2. Таймер, показывающий время до конца тестирования;
3. Поле, в котором отображается изображение сгенерированной формулы в математическом виде;
4. Текстовое поле для ввода формулы в текстовом виде;
5. Таблица, показывающая прогресс прохождения тестирования;
6. Кнопка «Показать подсказку», нажатие которой вызывает появление текста подсказки по вводу функций;
7. Кнопка «Следующее задание», при нажатии которой осуществляется переход к следующему заданию;
8. Кнопка «Прервать тестирование», нажатие которой инициирует завершение теста.

При нажатии на кнопку показа подсказки в окне появляется дополнительный блок с текстом, содержащий подсказку по вводу функций.

Окно режима тестирования с блоком подсказки показано на рисунке 2.5.

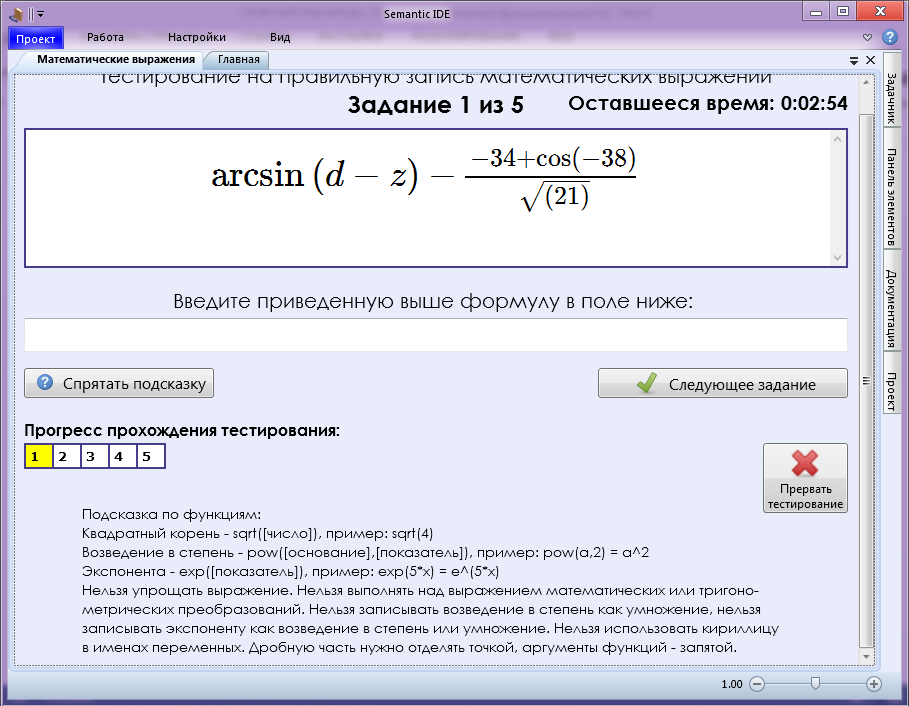


Рисунок 2.5 – Окно режима тестирования с блоком подсказки

При нажатии на кнопку перехода к следующему заданию выполняется проверка введенной в текстовый блок строки на соответствие заданной формуле, выполняется переход к следующему заданию, а таблица прогресса прохождения тестирования информирует пользователя о том, какие из пройденных заданий выполнены правильно.

На рисунке 2.6 показано окно режима тестирования после нескольких пройденных заданий.

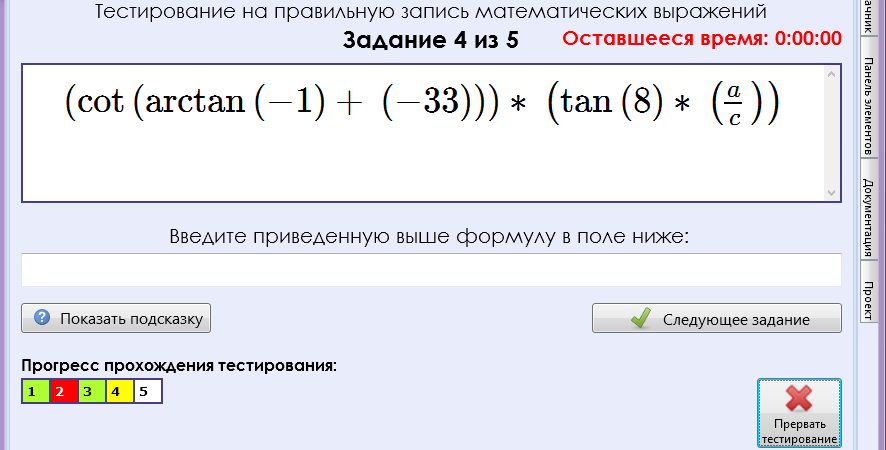


Рисунок 2.6 – Окно режима тестирования с информацией о пройденных заданиях

После выполнения всех заданий на экране показывается оценка и таблица с результатами тестирования, в которой для каждого задания показано, какое выражение было сгенерировано, какое введено пользователем, сколько времени пользователь затратил на выполнение этого задания, результат выполнение (правильно или неправильно) и пояснение к ошибке, если задание выполнено неправильно.

Окно с результатами тестирования показано на рисунке 2.7.



Рисунок 2.7 – Окно результатов тестирования с информацией о пройденных заданиях

При нажатии на кнопку прерывания тестирования, тест завершается, пройденные задания проверяются, а оставшиеся задания считаются выполненными неправильно.

Таблица результатов после прерывания тестирования показана на рисунке 2.8



Рисунок 2.8 – Окно результатов тестирования после прерывания теста

Если время, отведенное на тестирование, истекает раньше, чем пользователь справился со всеми заданиями, то пользователь получает уведомление о том, что время истекло и происходит прерывание теста аналогично тому, как это описано выше.

На рисунке 2.9 показано сообщение о том, что время на тест истекло.

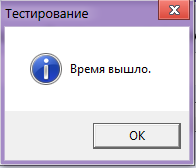


Рисунок 2.9 – Окно с сообщением об истекшем времени тестирования

* + 1. **Интерфейс окна настроек**

После нажатия кнопки «Настройки тестирования» в главном меню Semantic IDE пользователь попадает в окно настроек, показанное на рисунке 2.10.

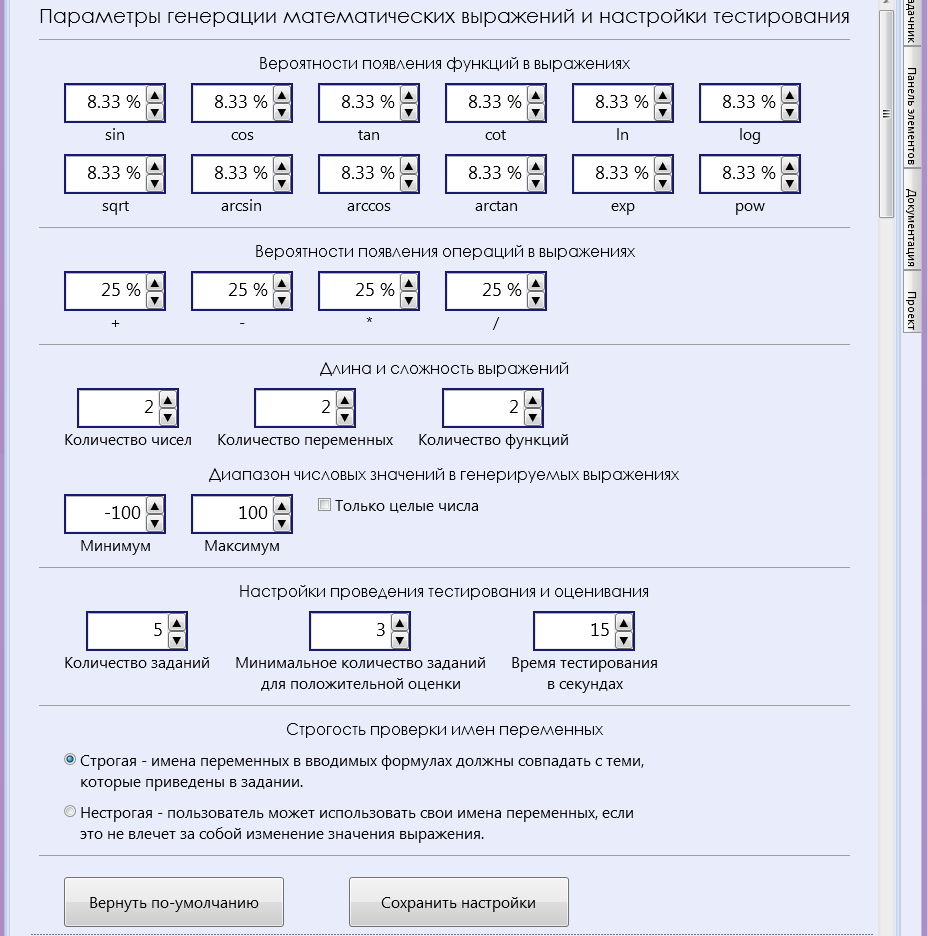


Рисунок 2.10 – Окно настроек

В окне настроек доступны следующие элементы управления:

1. Коллекция элементов для настройки вероятностей появления функций;
2. Коллекция элементов для настройки вероятностей появления операций;
3. Элементы для настройки длины и сложности выражений;
4. Элементы для настройки диапазона числовых значений;
5. Элементы для настройки проведения тестирования и оценивания;
6. Кнопки сохранения настроек и возвращения настроек по умолчанию.

Каждый элемент снабжен кнопками для увеличения и уменьшения значения этого элемента. Возможно также и ручное редактирование содержимого всех элементов, кроме содержащих значения вероятностей. После каждого изменения какой-либо из вероятностей, производится пересчет всех вероятностей, и они изменяются так, чтобы сохранялись их пропорции, и сумма вероятностей была равна 100%.

При нажатии на кнопку возвращения настроек по умолчанию, все значения элементов возвращаются в заранее заданное значение по умолчанию (например, вероятности появления всех функций устанавливаются в значение 8,333%, что делает их появления равновероятными).

При нажатии на кнопку сохранения настроек, выполняется проверка всех элементов на корректность содержащихся в них значений. Если некоторые из элементов содержат некорректные значения, пользователь получает сообщение о том, какие элементы заполнены неправильно, и сохранение не выполняется. Сообщение продемонстрировано на рисунке 2.11.

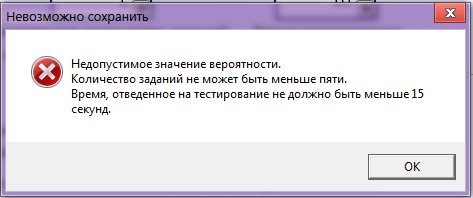


Рисунок 2.11 – Сообщение о невозможности сохранения настроек

Если все элементы заполнены корректно, то выполняется сохранение настроек. Сообщение об успешном сохранении продемонстрировано на рисунке 2.12.

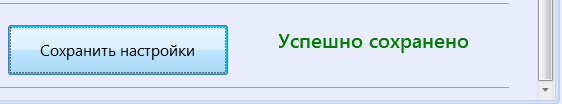


Рисунок 2.12 – Сообщение об успешном сохранении настроек

Настройки сохраняются в текстовый файл, поэтому при закрытии и повтором открытии среды настройки не теряются.

## 

# ПРИЛОЖЕНИЕ

**Блок-схемы и рисунки**



Рисунок П.1 – Блок-схема, демонстрирующая общий алгоритм работы программы