СОДЕРЖАНИЕ

[Введение 5](#_Toc421060917)

[1 Аналитический обзор литературы 7](#_Toc421060918)

[1.1 Анализ аналогичных программ 7](#_Toc421060919)

[1.1.1 Анализ программы «Notepad++» 7](#_Toc421060920)

[1.1.2 Анализ программы «Sublime Text» 8](#_Toc421060921)

[1.2 Анализ синтаксиса Object Pascal 9](#_Toc421060922)

[2 Разработка алгоритма 10](#_Toc421060923)

[2.1 Разработка общего алгоритма программы 10](#_Toc421060924)

[2.2 Схемы программы и предопределенных процессов 11](#_Toc421060925)

[3 Разработка программного средства 12](#_Toc421060926)

[3.1 Модуль «UnitMain» 12](#_Toc421060927)

[3.2 Модуль «StyleConfiguratorUnit» 12](#_Toc421060928)

[3.3 Модуль «XmlCodeUnit» 13](#_Toc421060929)

[3.4 Модуль «ErrorDisplayFormUnit» 13](#_Toc421060930)

[4 Обоснование технических приемов программирования 15](#_Toc421060931)

[4.1 Списки 15](#_Toc421060932)

[4.2 Деревья 15](#_Toc421060933)

[4.3 Текстовые файлы 16](#_Toc421060934)

[4.4 Отказ от использования компонента RichEdit 16](#_Toc421060935)

[4.5 Компонент Panel 16](#_Toc421060936)

[4.6 Компонент GroupBox 17](#_Toc421060937)

[4.7 Компонент ScrollBar 17](#_Toc421060938)

[4.8 Компонент ComboBox 17](#_Toc421060939)

[4.9 Компонент ListBox 17](#_Toc421060940)

[5 Тестирование программного средства 18](#_Toc421060941)

[6 Руководство пользователя 20](#_Toc421060942)

[Заключение 24](#_Toc421060943)

[Список использованных источников 25](#_Toc421060944)

[Приложение А. Исходный код программы 26](#_Toc421060945)

# Введение

Текстовый редактор — самостоятельная компьютерная программа или компонент программного комплекса (например, редактор исходного кода интегрированной среды разработки или окно ввода в браузере), предназначенная для создания и изменения текстовых данных в общем и текстовых файлов в частности.

Текстовые редакторы предназначены для работы с текстовыми файлами в интерактивном режиме. Они позволяют просматривать содержимое текстовых файлов и производить над ними различные действия — вставку, удаление и копирование текста, контекстный поиск и замену, сортировку строк, просмотр кодов символов и конвертацию кодировок, печать и т. п.

Часто интерактивные текстовые редакторы содержат дополнительную функциональность, призванную автоматизировать действия по редактированию (от записываемых последовательностей нажатий клавиш до полноценных встроенных языков программирования), или отображают текстовые данные специальным образом (например, с подсветкой синтаксиса).

Многие текстовые редакторы являются редакторами исходного кода, то есть они ориентированы на работу с текстами программ на тех или иных компьютерных языках.

На сегодняшний день существует множество текстовых редакторов, находящихся в свободном доступе. Все они предоставляют базовую возможность ввода и сохранения текста, а также позволяют настраивать параметры вывода текста, т.е. задавать шрифт, размер символов и стилистические модификаторы.

Если оценить статистику использования текстовых редакторов, то можно сделать однозначный вывод, что примитивные текстовые редакторы используются лишь в сфере информационных технологий.

Однако, если провести аналогию стороннего текстового редактора и полноценной среды разработки, предпочтение всегда отдается последнему, т.к. в средах разработки интегрированный текстовый редактор оснащен алгоритмом неоднозначной стилизации текста, «подсветкой синтаксиса», что многократно повышает удобочитаемость кода и, как следствие, ускоряет темпы разработки.

На сегодняшний день лидерами среди текстовых редакторов являются программы «Notepad++» и «Sublime Text», которые дают пользователю возможность гибко настраивать стиль и содержат сценарии для обработки кода, написанного на разных языках программирования.

В данном курсовом проекте передо мной встала задача проектирования текстового редактора с подсветкой синтаксиса и проверкой основных лексических структур для языка программирования Object Pascal. Целью работы является ознакомление с алгоритмами стилизации текста и проверки синтаксиса. Немалой частью проектирования программного средства является разработка дружелюбного и интуитивно понятного пользовательского интерфейса.

Для решения данной задачи мне необходимо провести аналитический обзор литературы, провести исследование аналогов, разработать алгоритм «умной» подсветки синтаксиса, который позволил бы работать с текстовым файлов максимально удобно, без простоев производительности при вводе текста, организовать независимое хранение множества текстовых файлов в памяти программы и корректное ее освобождение. Так же необходимо реализовать алгоритм базовой проверки синтаксиса.

Для систематизации полученных знаний, их корректировки и преобразования в нужную форму необходимо разработать алгоритмы данного программного средства и его отдельных модулей.

Подвергнуть отдельные модули программного средства тщательному тестированию для выявления слабых мест программы и улучшить алгоритмы.

Для готового программного средства будет необходимо написать руководство пользователя, в котором будут объясняться правила работы с программой.

# 1 Аналитический обзор литературы

## 1.1 Анализ аналогичных программ

### 1.1.1 Анализ программы «Notepad++»

Ярким примером текстового редактора с подсветкой синтаксиса является программа «Notepad++».

Функционал программы позволяет независимо открывать множество файлов, как текстовых, так и бинарных. Предоставляется выбор способа интерпретации текста как кода, написанного на одном из популярных языков программирования, таких как C++, C#, Java, Pascal, PHP и т.д.

Присутствует возможность гибкой настройки интерфейса пользователя, стилей подсветки синтаксиса, размера табуляции и макросов.

Текстовый редактор реализовывает функцию автозавершения кода, т.е. предлагает окончания лексических структур исходя из первых символов записи. Эта функция является крайне удобной при работе с HTML и CSS, где присутствуют лишь ключевые слова узлов и текст, что многократно ускоряет темпы веб-разработки и верстки.

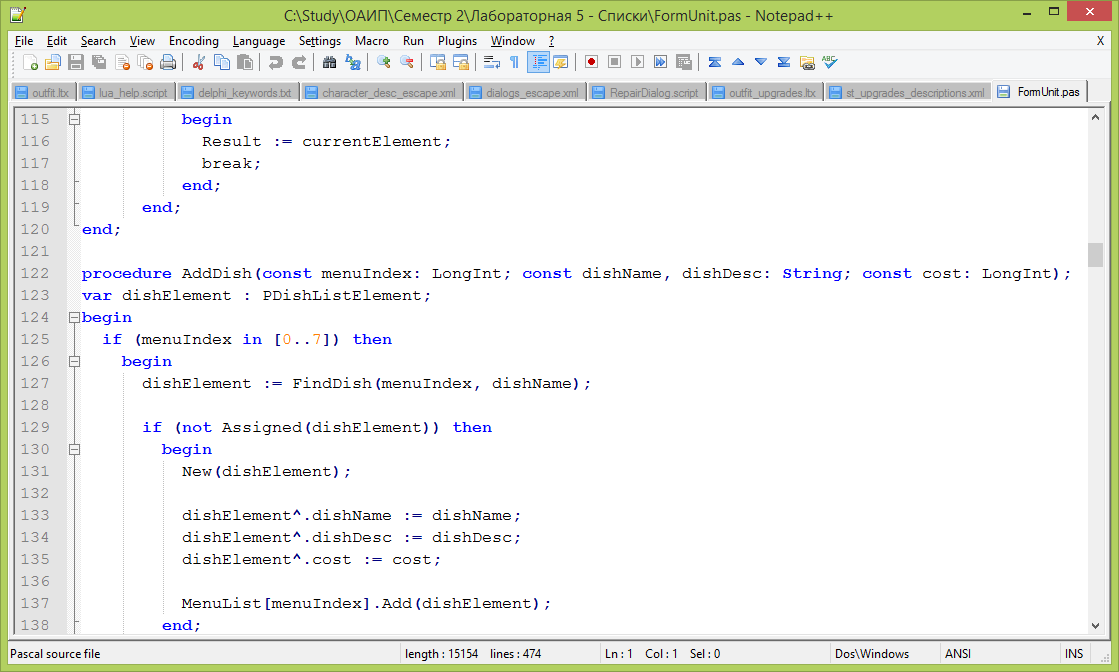


Рисунок 1.1 ­­– Приложение «Notepad++»

### 1.1.2 Анализ программы «Sublime Text»

Главным соперником текстового редактора «Notepad++» является новый текстовый редактор «Sublime Text».

**Достоинства:**

1. Поддерживает больше языков программирования;
2. Получает информацию об установленных на компьютере компиляторах и позволяет компилировать проекты прямо внутри текстового редактора;
3. Предоставляет расширенные возможности редактирования подсветки синтаксиса пользователем посредством записи регулярных выражений;
4. Реализует класс некоторых комбинаций клавиш, ускоряющих обработку текста.

**Недостатки:**

1. Бесплатный пакет приложения практически не содержит установленных плагинов;
2. Множество стандартных функций имеют только точки входа без реализации. Реализация осуществляется посредством скачивания соответствующего плагина из сети Интернет.

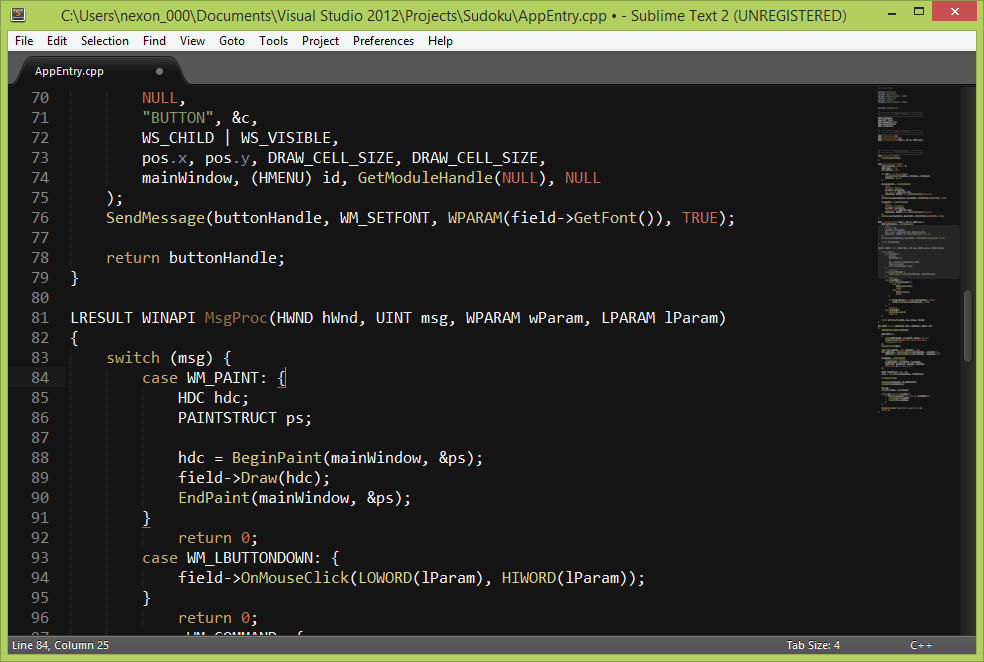


Рисунок 1.2 ­­– Приложение «Sublime Text 2»

## 1.2 Анализ синтаксиса Object Pascal

*Синтаксис языка программирования* — набор правил, описывающий комбинации символов алфавита, считающиеся правильно структурированной программой (документом) или её фрагментом. Синтаксису языка противопоставляется его семантика. Синтаксис языка описывает «чистый» язык, в то же время семантика приписывает значения (действия) различным синтаксическим конструкциям.

Каждый язык программирования имеет синтаксическое описание. Синтаксис языка можно описать, например, с помощью правил Бэкуса — Наура.

Синтаксис проверяется на ранних стадиях трансляции. В интерпретируемых языках программирования проверка синтаксиса производится или в процессе интерпретации (выполнения), или в процессе предварительной компиляции в промежуточный код. Кроме того, синтаксис может проверяться непосредственно при редактировании исходных текстов программ при использовании интегрированной среды разработки.

В процессе подготовки к выполнению задания, был детально изучен синтаксис языка программирования Object Pascal. Для изучения использовалось методическое пособие «Объектно-ориентированное программирование» авторов А.Н. Вальвачева, К.А. Суркова, Д.А. Суркова, Ю.М. Четырько.

Был уточнен полный список зарезервированных слов языка программирования.

Были изучены правила записи логических структур:

1. Секция объявления программы;
2. Секция подключения модулей;
3. Секция объявления пользовательских типов;
4. Секция констант;
5. Секция переменных;
6. Секция описания класса;
7. Объявление процедур и функций;
8. Правила передачи параметров в процедуры и функции, и модификаторы, которые могут применяться при передаче;
9. Описание составных операторов;
10. Описание условных операторов;
11. Описание операторов цикла:
12. Цикл с предусловием;
13. Цикл с постусловием;
14. Цикл с фиксированным количеством шагов.

# 2 Разработка алгоритма

## 2.1 Разработка общего алгоритма программы

Интерфейс программного средства должен соответствовать требованиям к текстовому редактору.

Программа должна предоставлять базовые функции для работы с текстом, интуитивно понятное окно стилизации текста, подсветку синтаксиса языка программирования Object Pascal, а также проверку синтаксиса в документе.

Сначала я разработал общий вид меню и панели инструментов редактора. Затем был тщательно изучен компонент RichEdit среды Delphi, так как он рассматривался в качестве компонента, позволяющего задавать стилизацию отдельных фрагментов текста. В последствии от этого компонента пришлось отказаться и написать свой собственный компонент на основе другого (см. пункт 4.4).

Исходя из этого был подготовлен список функций, который должен быть реализован в программе:

1. Загрузка и сохранение файлов
2. Копирование и вставка текстовых данных из буфера обмена
3. Реализация алгоритма подсветки синтаксиса.
4. Предоставление пользователю возможности гибко настраивать стиль оформления текста
5. Реализация алгоритма проверки синтаксиса в документе

## 2.2 Схемы программы и предопределенных процессов



Рисунок 2.1 – Схема работы программы

# 3 Разработка программного средства

## 3.1 Модуль «UnitMain»

Главный модуль программы, который осуществляет реализацию главного интерфейса текстового редактора.

Предоставляет пользователю набор функций для работы с файлами:

1. Загрузка текстового файла;
2. Сохранение текущего открытого в программе текстового файла;
3. Закрытие текущего открытого файла и освобождение занятой им памяти.

Предоставляет пользователю стандартный набор функций для обработки текста:

1. Копирование;
2. Вставка;
3. Удаление;
4. Перемещение по текстовому полю и стандартные клавишные комбинации, которые являются стандартами при работе с текстом.

Модуль реализует класс TCodeTextField, наследованный от класса TImage. Конкретный класс TImage был выбран потому, что он обладает свойством Canvas, которое реализует удобный графический вывод в среде Delphi. Стандартный компонент RichEdit для целей, преследуемых в данном курсовом проекте, не подходит (см. пункт 4.4).

Так же модуль предоставляет реализацию дополнительного класса TLinesCounter, который используется в связке с классом TCodeTextField. Данных класс представляет собой вертикальную полосу с нумерацией строк текстового поля.

Класс TCodeTextField реализует алгоритм подсветки синтаксиса. Код «подсвечивается» полностью при загрузке файла в текстовое поле, а затем подсветка текста изменяется как реакция на ввод пользователя, что не влечет за собой полный перерасчет подсветки, а лишь области, с которой провзаимодействовал пользователь.

Предоставляет пользователю кнопки для перехода в окно редактирования стиля оформления текстового редактора и окно проверки синтаксиса открытого документа.

Так же пользователь может просмотреть краткую справочную информацию о программном средстве и разработчике.

## 3.2 Модуль «StyleConfiguratorUnit»

Модуль используется для задания стиля оформления отдельных элементов текстового документа при подсветке синтаксиса.

Модуль позволяет изменять такие элементы стиля, как:

1. Цвет текста;
2. Цвет фона;
3. Семейство шрифтов;
4. Размер символа;
5. Модификатор стиля (полужирный, курсив, подчеркнутый).

Выбранные настройки стиля можно отменить.

При нажатии кнопки «Применить» настройки стиля сохраняются в оперативной памяти для единичного использования, следовательно настройки сбросятся при завершении работы программы. Так же при нажатии кнопки «Сохранить», выбранные настройки сохранятся в файл на диске, что позволит восстановить их при последующих запусках программы.

## 3.3 Модуль «XmlCodeUnit»

Модуль содержит функции для построения объектной модели документа.

Построение объектной модели производится в несколько этапов:

1. Разбиение кода документа на составные части, разделенные ключевыми словами языка Object Pascal, найденными в процессе проверки;
2. Поэтапная проверка полученных строковых выражений в соответствии с последним ключевым словом, которое задает поведение обработчика выражений;
3. При смене текущего ключевого слова изменяется поведение обработчика, и если смена исключается в виду несостоятельности логической последовательности документа, записанного на языке программирования Object Pascal, сообщение об этом выводится в лог проверки синтаксиса.

## 3.4 Модуль «ErrorDisplayFormUnit»

Модуль предоставляет пользователю форму с отчетом о проверке синтаксиса документа.

Пользователь может получить информацию о названии файла, где произошла ошибка, номер строки, в которой произошла информация, и тип ошибки, определенный программой.

Программное средство определяет такие ошибки синтаксиса, как:

1. Неверное количество скобок;
2. Закрывающая скобка раньше открывающей;
3. Ошибка в объявлении параметров процедуры;
4. Некорректное название процедуры;
5. Некорректное имя идентификатора;
6. Некорректное название переменной;
7. Ожидалось объявление переменной;
8. Ошибка в объявлении переменной;
9. Ожидалось объявление константы;
10. Некорректное имя типа;
11. Некорректный диапазон;
12. Правая граница диапазона меньше левой;
13. Ошибка в объявлении цикла FOR;
14. Некорректное выражение;
15. Ожидалось определение функции;
16. Ожидалось определение процедуры;
17. Ожидалось условие.

# 4 Обоснование технических приемов программирования

## 4.1 Списки

Список — это абстрактный тип данных, представляющий собой упорядоченный набор значений, в котором некоторое значение может встречаться более одного раза. Экземпляры значений, находящихся в списке, называются элементами списка; если значение встречается несколько раз, каждое вхождение считается отдельным элементом.

Существует два вида списков:

1. Связные списки
2. Списки на основе массивов

Связный список отличается гибкостью структуры: он легко поддается редактированию, добавлению и удалению элементов, однако в таком списке невозможно добиться прямого доступа к произвольному элементу списка.

Список на основе массива имеет произвольный доступ к любому элементу через указатель и смещение, так же легок в редактировании элемента. Однако при добавлении к такому списку элемента, весь массив необходимо копировать в новую область памяти, что занимает значительное время при перемещении объемных данных.

Исходя из задачи, поставленной в данном курсовом проекте, было решено использовать списки на основе массивов, так как именно эти списки дают произвольный доступ к элементам, что в работе требуется гораздо чаще, чем редактирование списков.

## 4.2 Деревья

Дерево — одна из наиболее широко распространённых структур данных в информатике, эмулирующая древовидную структуру в виде набора связанных узлов. Является связанным графом, не содержащим циклы. Большинство источников также добавляют условие на то, что рёбра графа не должны быть ориентированными. В дополнение к этим трём ограничениям, в некоторых источниках указываются, что рёбра графа не должны быть взвешенными.

В курсовом проекте дерево используется для поиска ключевых слов в реализации алгоритма подсветки синтаксиса. Такое решение было принято в следствие изучения стандартной строковой функции Pos, которая из-за своей реализации оказалась несостоятельной и неудобной в использовании при поиске экземпляров объектов внутри одной строки. Реализация поиска по дереву осуществляет один проход по тексту и обнаруживает все ключевые слова за минимальное количество операций, и, как следствие, ускоряет скорость работы алгоритма поиска многократно.

## 4.3 Текстовые файлы

Текстовые файлы обеспечивают простой и удобный способ сохранения текстовых данных. Они обеспечивают простой механизм для чтения и записи численных данных, сохраненных как текст.

Так как тестовый редактор не подразумевает редактирование бинарных данных, привычного текстового файла оказывается достаточно для выполнения курсового проекта. Текстовый файл предоставляет удобные интерфейсы для чтения файла с физического носителя и записи файла на физический носитель, так как коды символов внутри файла находятся в диапазоне значений, которые имеют символьную презентацию в системе.

Для выполнения данного курсового проекта текстовый файл является лучшим решением.

## 4.4 Отказ от использования компонента RichEdit

В процессе оценки компонентов для реализации текстового поля программы, был выбран компонент RichEdit. Этот компонент допускает стилизацию отдельных фрагментов текста. Однако, этот компонент оказался несостоятельным при подсветке синтаксиса в виду нескольких причин:

1. Для выделения фрагмента текста внутри RichEdit необходимо задать позицию начала выделения и длину выделения. При этом фрагмент текста выделяется визуально, т.е. каретка на некоторый момент перемещается к месту выделения, и выделенный фрагмент выделяется противоположным цветом. Этот визуальный недостаток невозможно исправить стандартными возможностями Delphi;
2. При выделении полужирным начертанием фрагментов текста, остальные фрагменты получают некоторое горизонтальное и вертикальное смещение в зависимости от конкретных символов. Стандартными средствами компонента нельзя добиться строгого выравнивания текста.

Исходя из этого, было принято решение о написании класса, который будет унаследован от класса TImage. Это позволило произвести гибкую настройку визуализации текста и исправить критичные недостатки, обнаруженные в компоненте RichEdit.

## 4.5 Компонент Panel

Панели Panel наиболее широко используются в приложениях. С их помощью группируются различные элементы интерфейса (кнопки, окна редактирования, списки), функционально связанные друг с другом. Такая функциональная связь должна поддерживаться и зрительной связью — объединением соответствующих элементов в рамках одной панели. Панели Panel могут также использоваться для организации инструментальных панелей, полос состояния и т.п.

В данном проекте компонент Panel используется для создания панели инструментов, на которой размещены кнопки, дублирующие пункты меню. Каждая кнопка оснащена прикрепленной картинкой, которая достаточно наглядно объясняет назначение кнопки.

## 4.6 Компонент GroupBox

Панель GroupBox не имеет таких широких возможностей задания различных стилей оформления, как Panel. Но она имеет встроенную рамку с надписью, которая обычно используется для выделения на форме группы функционально объединенных компонентов. Никаких особых свойств, отличных от уже рассмотренных, панель GroupBox не имеет.

## 4.7 Компонент ScrollBar

Компонент Delphi ScrollBar — элемент управления в виде ползунка. Представляет собой горизонтальную или вертикальную полосу прокрутки.

В данном проекте используются два таких компонента, один для вертикальной прокрутки текстового поля, другой — для горизонтальной.

## 4.8 Компонент ComboBox

Компонент Delphi ComboBox представляет собой комбинацию списка строк ListBox со строкой ввода Edit. При этом "список строк" компонента Delphi ComboBox вначале скрыт, и раскрывается при щелчке мышкой по треугольнику раскрытия, который находится справа в строке ввода.

Таким образом, с помощью Delphi ComboBox место на форме экономится для размещения других элементов интерфейса программы.

В курсовом проекте этот компонент используется для выбора шрифта текстового поля. Компонент ComboBox удобен, если пользователю требуется выбрать один из предложенных вариантов, и, если вариантов много, он занимает минимум места.

## 4.9 Компонент ListBox

Компонент ListBox — это компонент, служащий для хранения и обработки текстовой информации. Каждая строка ListBox хранит строку данных в формате String.

Компонент ListBox является массивом строк. ListBox может загружать данные с диска, и сохранять информацию в файл. Также ListBox может сортировать строки. Доступ к строкам предоставляет свойство Items этого компонента.

В проекте компонент ListBox используется при выборе элемента интерфейса в окне стилизации документа.

# 5 Тестирование программного средства

Тестирование программного средства представлено в виде таблицы (Таблица 5.1), в которую включены: действие пользователя, ожидаемый результат и реальный результат (на основании которого делается вывод, пройден ли тест).

В ходе тестирования программного средства необходимо проверить работоспособность отдельных процедур и функций в различных ситуациях. Главной задачей тестирования является выявление слабых мест программы для последующего устранения неполадок системы.

|  |  |  |
| --- | --- | --- |
| Действие пользователя | Ожидаемый результат | Реальный результат |
| Ввод текста в текстовое поле | Обновление подсветки синтаксиса | Тест пройден |
| Ввод некорректного имени типа (функция IsCorrectTypeName) | Вывод ошибки «Некорректное имя типа» | Тест пройден |
| Ввод некорректного выражения (функция IsCorrectStatement) | Вывод ошибки «Некорректное выражение» | Тест пройден |
| Изменение настроек стиля | Применение новых настроек к активному текстовому документу | Тест пройден |
| Нажатия кнопки «Вырезать» на панели инструментов | Выделенный текст копируется в буфер обмена, выделенный фрагмент из текста удаляется | Тест пройден |
| Нажатие комбинации клавиш Ctrl+X | Происходит обращение к функции «Вырезать» | Тест пройден |
| Нажатие кнопки «Проверка синтаксиса» | Открывается форма, содержащая текстовую интерпретацию дерева объектной модели документа | Тест пройден |
| Нажатие кнопки «Просмотреть статус проверки» | Пользователь видит статус проверки синтаксиса и описания всех ошибок, обнаруженных при проверке | Тест пройден |
| Нажатие кнопки «Вставить» | Если в буфере обмена находятся данные текстового типа, они добавляются в текущую позицию каретки текстового поля | Тест пройден |
| Нажатие кнопки «Новый файл» | Создается новая вкладка файла с названием «new.txt» | Тест пройден |
| Нажатие кнопки «О программе» | Выводится информация о разработчике программы | Тест пройден |
| Ввод названия переменной начинается с цифры | Вывод ошибки «Неверное имя идентификатора» | Тест пройден |
| Повторное объявление секции «Program» внутри документа | Вывод ошибки «Повторное объявление секции PROGRAM» | Тест пройден |
| Ответ «Нет» в окне предупреждения о закрытии программы | Программа не закрывается, пользователь имеет возможность сохранить документы, в которых были сделаны изменения | Тест пройден |

Таблица 5.1 – Результаты тестирования

В результате тестирования ошибок выявлено не было, все тесты были пройдены.

# 6 Руководство пользователя

Интерфейс данной программы интуитивно понятен. В главное меню вынесены кнопки, отвечающие за обобщенный функционал программного средства.

Секция меню «Файл» предоставляет пользователю возможность создать новый файл, открыть уже существующий, сохранить отредактированный файл, открытый в программе в данный момент, закрыть файл и выйти из программы.

Секция меню «Правка» предоставляет пользователю возможность скопировать, вырезать и вставить текст, выделить весь текст, а также удалить символ справа от каретки.

Секция меню «Настройки» содержит единственный пункт «Стиль оформления». Выбрав в этот пункт, программа покажет вам форму и предложит выбрать параметры стиля для текста и подсветки синтаксиса.

В меню «Помощь» пользователь может увидеть информацию о разработчике программного средства.

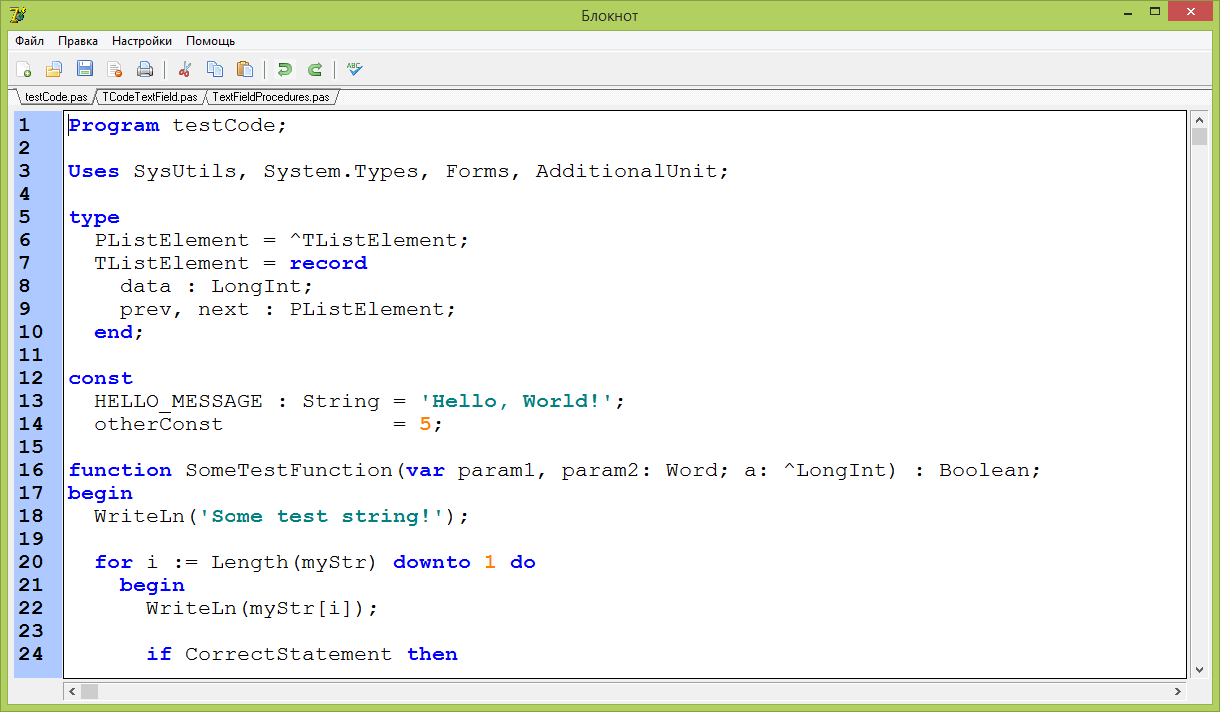


Рисунок 6.1 ­­– Интерфейс программы

Панель инструментов содержит дубликаты функций, доступных в главном меню программы. За исключением последней кнопки, крайней справа, которая осуществляет проверку синтаксиса кода, записанного в открытом для редактирования файле.



Рисунок 6.2 ­­– Панель инструментов

Под панелью инструментов находится панель вкладок, содержащая названия открытых в течение выполнения программы файлов.



Рисунок 6.3 ­­– Список открытых файлов

Далее располагается область редактирования текстового документа. Документ интерпретируется как код, написанный на языке Object Pascal. Окно редактирования текста обрамлено двумя полосами прокрутки, вертикальной и горизонтальной. Вертикальная полоса прокрутки используется для перемещения между строками документа, а горизонтальная используется для передвижения по строке.

Слева от окна редактирования текста располагается компонент, отображающий нумерацию строк текста.

При переходе в меню «Настройки» ­­– «Стиль оформления», пользователь увидит форму, предлагающую выбрать элемент стиля, который пользователь хотел бы настроить, и выбрать цвет текста, цвет фона и шрифт для заданного элемента стиля.

После того, как стиль оформления установлен, можно применить стиль, в таком случае он сразу же применится к открытому текстовому документу. Если нажать «Отмена», стиль оформления сбросится.

При нажатии кнопки «Сохранить», настройки стиля сохраняются в сторонний файл, который загружается при запуске программы.

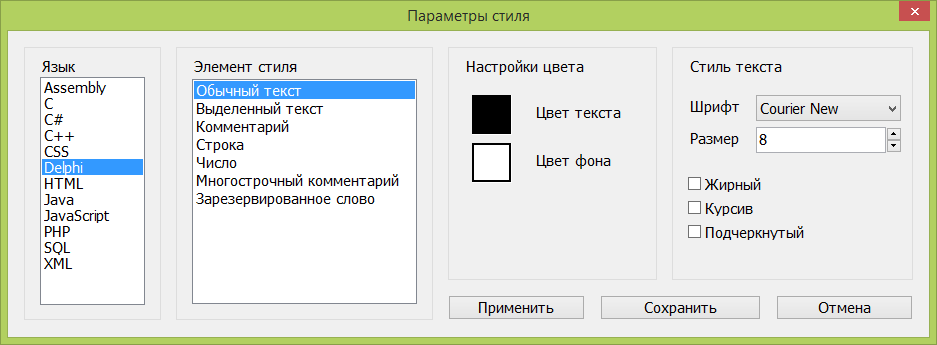


Рисунок 6.4 ­­– Окно изменения параметров стиля

При нажатии на кнопку проверки синтаксиса на панели инструментов, пользователь видит форму, в которой строится объектная модель документа (DOM – Document Object Model). Эта модель визуализируется с помощью текстового XML–дерева. Объектная модель разбивает документ на составные части независимо от стиля написания кода конкретным пользователем, модель является универсальной.

Базовыми задачами программного средства являются подсветка и проверка синтаксиса языка программирования Object Pascal, однако объектная модель документа может расширить возможности проверки, добавляя потенциальную возможность проверки логики программы, а также портирования кода из одного объектно-ориентированного языка программирования в другой, если во фрагменте кода не присутствует языковых особенностей конкретных языков.

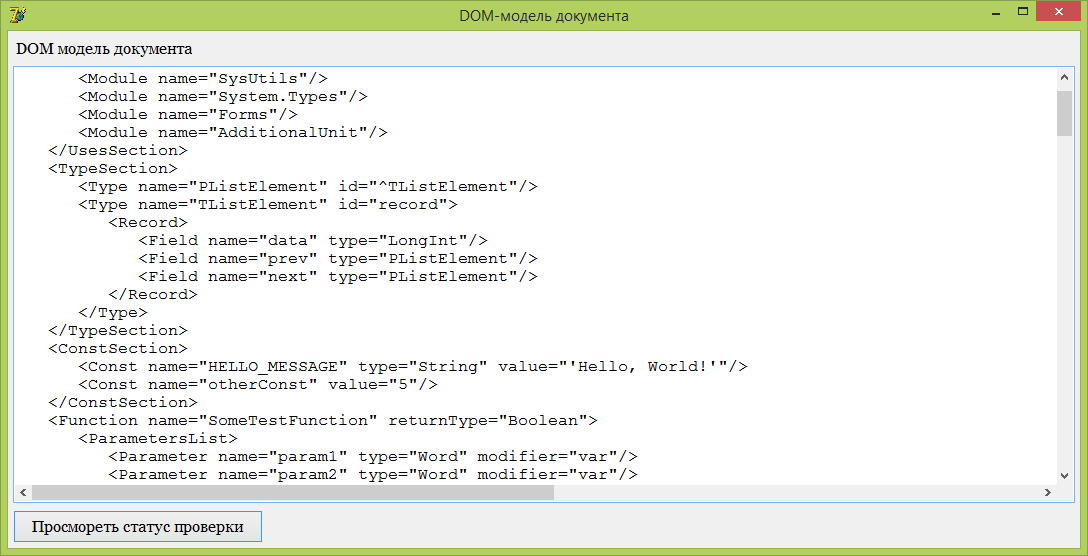


Рисунок 6.5 ­­– Форма представления объектной модели документа

Нажав на кнопку «Просмотреть статус проверки», пользователь увидит окно, содержащее результат проверки синтаксиса в документе.

В результате проверки, каждая найденная ошибка выводится в отдельной строке.

Строка ошибки содержит базовую информацию о найденной ошибке:

1. Название файла, в котором найдена ошибка;
2. Номер строки кода, в которой найдена ошибка;
3. Текстовое описание ошибки

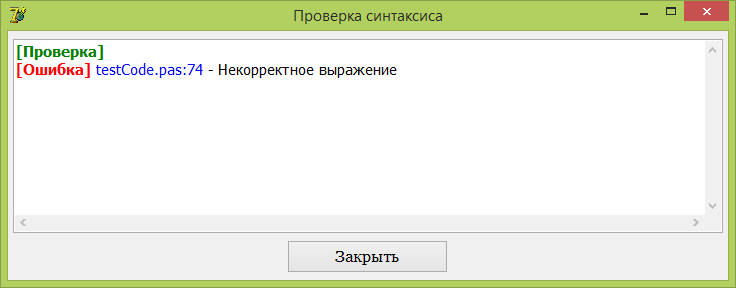


Рисунок 6.6 ­­– Результат проверки синтаксиса

При нажатии на кнопку «Помощь» – «О программе», пользователь увидит краткую справочную информацию о создателе программного средства.

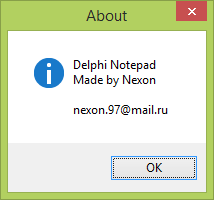


Рисунок 6.7 – Окно информации о программе

При закрытии окна приложения пользователь увидит предупреждение о том, что все файлы нужно сохранять вручную.

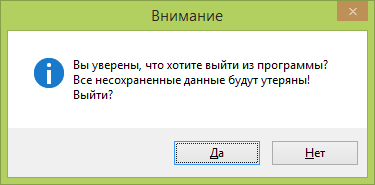


Рисунок 6.8 – Предупреждении о закрытии

# Заключение

В ходе создания текстового редактора с подсветкой и проверкой синтаксиса мною была изучена соответствующая литература, выполнен поиск существующих аналогов и их обзор. После чего были ясно сформулированы задачи данного курсового проекта, найдены пути их решения.

Я однозначно определил пользовательский интерфейс программы и базовый функционал, который должен быть реализован в текстовом редакторе.

Далее утвердил структуру программы, разработал алгоритм связывания функциональных блоков программного средства и его схему. Выбрав наиболее подходящие для ситуации технические методы решений мною было создано программное средство, обладающее требуемым функционалом. В дальнейшем оно было испытанно на различный ввод информации и корректное поведение в нестандартных ситуациях.

После этапа тестирования было составлено подробное руководство пользователя.

В конечном счете я получил программное средство, отвечающее заданным требованиям, способным выполнять поставленные перед ним задачи, с интуитивно понятным интерфейсом, базовым функционалом и возможностью корректно обрабатывать различные ситуации.

# Список использованных источников

[1] Глухова Л.А. Учебное пособие по дисциплине «Основы алгоритмизации и программирования» - Минск, 2006. – 195 стр.

[2] Справочник – «Основы Delphi» [Электронный ресурс] – Режим доступа: http://www.delphibasics.ru/ - Дата доступа: 27.03.2015

[3] С. Макконнелл «Совершенный код» - Москва, 2014 г. – 896 стр.

# Приложение А. Исходный код программы