**СОДЕРЖАНИЕ**

1. Введение.........................................................................................................3
2. Обзор методов и алгоритмов поставленной задачи...................................5

2.1 Библиотеки для создания графических оболочек................................5

2.1.1 QT.................................................................................................5

2.1.2 Windows Forms...........................................................................6

2.1.3 Microsoft Foundation Classes......................................................7

2.1.4 С++ Builder VCL.........................................................................8

2.2 Редактирование кода.............................................................................10

2.2.1 Редактирование в текстовом виде..........................................10

2.2.2 Визуальный редактор..............................................................10

3. Обоснование выбранных методов и алгоритмов........................................11

4.Описание программы для программиста......................................................13

4.1 Диаграмма классов................................................................................13

4.2 Описание классов…………………......................................................13

5. Описание алгоритмов решения задачи..........................................................21

6. Руководство пользователя..............................................................................25

7.Заключение........................................................................................................27

8. Список литературы..........................................................................................28

**ВВЕДЕНИЕ**

Язык C++ официально получил свое название в 1983 г. Он был создан на основе более старого языка Си и имел целью упростить процесс создания программ. C++ позволил программистам составлять алгоритмы с помощью привычных общечеловеческих понятий. При этом C++ сохраняет преимущества Си и позволяет добиться весьма высокого быстродействия получаемых программ. Сегодня язык C++ очень широко распространен во всем мире. Большинство программ, как в России, так и за рубежом создают именно на этом языке. В частности, операционная система Windows написана средствами языка C++.

C++ поддерживает такие парадигмы программирования как процедурное программирование, модульность, раздельная компиляция, обработка исключений, абстракция данных, типы (объекты), виртуальные функции, объектно-ориентированное программирование, обобщенное программирование, контейнеры и алгоритмы, сочетает свойства как высокоуровневых, так и низкоуровневых языков. Наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

C++ Builder — программный продукт, инструмент быстрой разработки приложений (RAD), интегрированная среда программирования (IDE), система, используемая программистами для разработки программного обеспечения на языке программирования C++. Изначально разрабатывался компанией Borland Software, а затем её подразделением CodeGear, ныне принадлежащим компании Embarcadero Technologies. C++ Builder объединяет в себе комплекс объектных библиотек (STL, VCL, CLX, MFC и др.), компилятор, отладчик, редактор кода и многие другие компоненты. Большинство компонентов, разработанных в Delphi, можно использовать и в C++ Builder без модификации, но обратное утверждение не верно.

C++ Builder содержит инструменты, которые при помощи drag-and-drop действительно делают разработку визуальной, упрощает программирование благодаря встроенному WYSIWYG — редактору интерфейса и пр. Кроме того в C++ Builder встроены такие полезные библиотеки С++ как STL и Boost.

При написании курсовой работы использовалась версия RAD Studio XE2 выпущенная в 2011, которая позволяет писать высокопроизводительные приложения для Windows и Mac OS для 32 битных и 64 битных систем.

**2 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ ПОСТАВЛЕННОЙ ЗАДАЧИ**

**2.1 Библиотеки для создания графических оболочек**

**2.1.1 Qt**

Qt – [кросс-платформенный](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) инструментарий разработки ПО на языке программирования [C++](http://ru.wikipedia.org/wiki/C%2B%2B). Позволяет запускать написанное с его помощью [ПО](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) в большинстве современных [операционных систем](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) путём простой компиляции программы для каждой ОС без изменения [исходного кода](http://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4). Включает в себя все основные [классы](http://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), которые могут потребоваться при разработке [прикладного программного обеспечения](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), начиная от элементов [графического интерфейса](http://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) и заканчивая классами для работы с [сетью](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C), [базами данных](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и [XML](http://ru.wikipedia.org/wiki/XML).

Начиная с версии 4.5 Qt распространяется по 3 лицензиям (независимо от лицензии, исходный код Qt один и тот же):

* [Qt Commercial](http://ru.wikipedia.org/wiki/Qt_Commercial) – для разработки ПО с собственнической лицензией, допу-скающая модификацию самой Qt без раскрытия изменений;
* [GNU GPL](http://ru.wikipedia.org/wiki/GNU_GPL) – для разработки ПО с открытыми исходными кодами распространяемыми на условиях [GNU GPL](http://ru.wikipedia.org/wiki/GNU_GPL);
* [GNU LGPL](http://ru.wikipedia.org/wiki/GNU_LGPL) – для разработки ПО с собственнической лицензией, но без внесения изменений в Qt.

Со времени своего появления в [1996 году](http://ru.wikipedia.org/wiki/1996_%D0%B3%D0%BE%D0%B4) библиотека Qt легла в основу тысяч успешных проектов во всём мире. Кроме того, Qt является фундаментом популярной рабочей среды [KDE](http://ru.wikipedia.org/wiki/KDE), входящей в состав многих [дистрибутивов](http://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D1%82%D1%80%D0%B8%D0%B1%D1%83%D1%82%D0%B8%D0%B2_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B) [Linux](http://ru.wikipedia.org/wiki/Linux" \o "Linux).

Отличительная особенность Qt от других библиотек — использование [Meta Object Compiler](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80" \o "Метаобъектный компилятор) (MOC) — предварительной системы обработки исходного кода. MOC позволяет во много раз увеличить мощь библиотек, вводя такие понятия, как слоты и сигналы. Кроме того, это позволяет сделать код более лаконичным. Утилита MOC ищет в заголовочных файлах на [C++](http://ru.wikipedia.org/wiki/C%2B%2B) описания классов, содержащие макрос Q\_OBJECT, и создаёт дополнительный исходный файл на [C++](http://ru.wikipedia.org/wiki/C%2B%2B), содержащий метаобъектный код.

Недостаток - вынужденное оперирование с самым высоким "слоем" операционной системы, без использования полезных, но специфических для каждой ОС архитектурных возможностей.

**2.1.2 Windows Forms**

Windows Forms — название интерфейса программирования приложений (API), отвечающего за графический интерфейс пользователя и являющегося частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обертки для существующего Win32 API в управляемом коде. Причем управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, С++, так и на VB.Net, J# и др.

С одной стороны Windows Forms рассматривается как замена более старой и сложной библиотеке MFC, изначально написанной на языке C++, но с другой стороны, WF не предлагает парадигму, сравнимую с MVC. Для исправления этой ситуации и реализации данного функционала в WF существуют сторонние библиотеки. Одной из наиболее используемых подобных библиотек является User Interface Process Application Block, выпущенная специальной группой Microsoft, занимающейся примерами и рекомендациями, для бесплатного скачивания. Эта библиотека также содержит исходный код и обучающие примеры для ускорения обучения.

Недостаток – требование наличия установленного .NET Framework и как следствие отсутсвие кроссплатформенности.

**2.1.3 Microsoft Foundation Classes**

Пакет Microsoft Foundation Classes (MFC) — библиотека на языке C++, разработанная Microsoft и призванная облегчить разработку GUI-приложений для Microsoft Windows путем использования богатого набора библиотечных классов.

Библиотека MFC, как и её основной конкурент, Borland VCL, облегчает работу с GUI путем создания каркаса приложения — «скелетной» программы, автоматически создаваемой по заданному макету интерфейса и полностью берущей на себя рутинные действия по его обслуживанию (отработка оконных событий, пересылка данных между внутренними буферами элементов и переменными программы и т. п.). Программисту после генерации каркаса приложения необходимо только вписать код в места, где требуются специальные действия. Каркас должен иметь вполне определенную структуру, поэтому для его генерации и изменения в Visual C++ предусмотрены мастера.

Кроме того, MFC предоставляет объектно-ориентированный слой оберток (англ. wrappers) над множеством функций Windows API, делающий несколько более удобной работу с ними. Этот слой представляет множество встроенных в систему объектов (окна, виджеты, файлы и т. п.) в виде классов и опять же берет на себя рутинные действия вроде закрытия дескрипторов и выделения/освобождения памяти.

Добавление кода приложения к каркасу реализовано двумя способами. Первый использует механизм наследования: основные программные структуры каркаса представлены в виде классов, наследуемых от библиотечных. В этих классах предусмотрено множество виртуальных функций, вызываемых в определенные моменты работы программы. Путем доопределения (в большинстве случаев необходимо вызвать функцию базового класса) этих функций программист может добавлять выполнение в эти моменты своего кода.

Второй способ используется для добавления обработчиков оконных событий. Мастер создает внутри каркасов классов, связанных с окнами, специальные массивы — карты (оконных) сообщений (англ. message map), содержащие пары «ИД сообщения — указатель на обработчик». При добавлении/удалении обработчика мастер вносит изменения в соответствующую карту сообщений.

Недостатки:

* Низкоуровневый и как следствие отсутствует кроссплатформенность
* Сложно расширять

**2.1.4 С++ Builder VCL**

Библиотека визуальных компонентов (англ. Visual Component Library, VCL) - объектно-ориентированная библиотека для разработки программного обеспечения, разработанная компанией Borland (на данный момент поддерживается Embarcadero) для поддержки принципов визуального программирования. VCL входит в комплект поставки Delphi, C++ Builder и Embarcadero RAD Studio и является, по сути, частью среды разработки, хотя разработка приложений в этих средах возможна и без использования VCL. VCL предоставляет огромное количество готовых к использованию компонентов для работы в самых разных областях программирования, таких, например, как интерфейс пользователя (экранные формы и элементы управления) работа с базами данных, взаимодействие с операционной системой, программирование сетевых приложений и прочее.

Уже умея программировать на Delphi легко прейти к работе в C++ Builder. Легко расширяется с помощью написания новых компонентов.

**2.2 Редактирование кода**

**2.2.1 Редактирование в текстовом виде**

Для редактирования HTML кода можно было бы использовать поле ввода типа Memo, редактирование кода и его форматирование можно осуществить с помощью строковых операций C/C++. Отображать результат можно в компонентах поддерживающих отображение HTML разметки.

Недостатки:

* Необходимость написать объекты для разбора HTML кода, т.к. надо знать каким образом отформатирован та или иная часть документа.
* Такой редактор нельзя назвать визуальным, так как нельзя непосредственно редактировать отображение в браузере, а только через код

**2.2.2 Визуальный редактор HTML-страницы**

Использование компонента операционной системы WebBrowser позволит работать с графическим видом HTML. Отображение HTML в виде кода потребуется только для тонкого изменения некоторых частей кода.

Недостатки:

* Потребуется работать в с низкоуровневым интерфейсов COM объекта.
* Необходимо наличия установленного Internet Explorer в системе

**3 ОБОСНОВАНИЕ ВЫБРАННЫХ МЕТОДОВ И АЛГОРИТМОВ**

Для написания приложения использовалась среда разработки C++ Builder RAD Studio XE2. Графический интерфейс реализован в виде библиотеки VCL так как необходимо одновременно реализовать красивый и функциональный интерфейс и получить доступ к низкоуровневому компоненту операционной системы - WebBrowser.

WebBrowser реализован в виде компонента TCppWebBrowser, который реализует простейшую функциональность Веб браузера. Для более сложного управления необходимо работать с интерфейсами COM (Component Object Model — объектная модель компонентов).

Для работы приложения потребуется установленный браузер Internet Explorer, это требование осуществляется во всех современных операционных системах Microsoft.

При выборе данного подхода нет необходимости писать сложные алгоритмы обработки HTML кода, так как эти алгоритмы уже реализованы в WebBrowser.

В то же время написание графического интерфейса займет меньше времени и меньше кода, чем написание кода напрямую через WIN API или MFC.

Для реализации многооконного итерфейса использован принцип написания многооконного интерфейса MDI.

Платой за простоту разработки является объем полученного исполняемого файла, который значительно превысит размер файла полученного при написании кода на чистом Windows API. Однако этот недостаток несущественен, учитывая объемы современных средств хранения данных.

**4. ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПРОГРАММИСТА**

**4.1 Диаграмма классов**

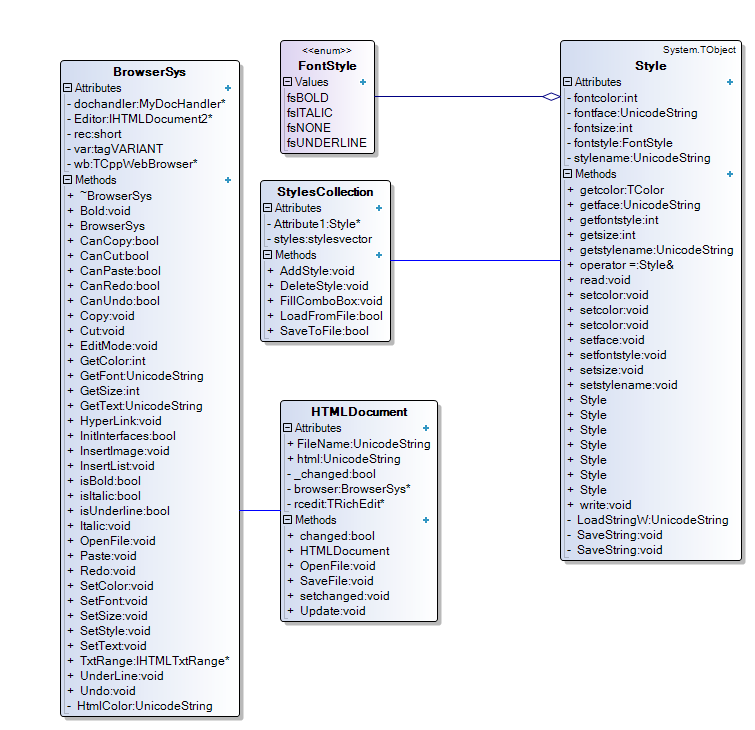
****

Рисунок 4.1 – Диаграмма классов

**4.2 Описание классов**

**4.2.1 Класс Style**

Класс Style представляет стиль форматирования теста, включает следующие атрибуты форматирования:

* Цвет текста (fontcolor)
* Шрифт (fontface)
* Размер текста (fontsize)
* Стиль теста (fontstyle)
* Имя стиля (stylename)

class Style : public TObject {

public:

Style(); // конструктор по умолчанию

Style(int size, int color, const char \*font, int fontstyle,

String stylename = ""); // первый конструктор

Style(int size, TColor color, const char \*font, int fontstyle,

String stylename = ""); // второй конструктор

Style(const Style &obj); // конструктор копирования

Style& operator = (const Style & right); // перегруженный оператор сравнения

void setsize(int); // Сеттеры и геттеры атрибутов

int getsize() const ;

void setface(String face);

String getface() const ;

void setcolor(int);

void setcolor(TColor);

void setstylename(String stylename);

String getstylename() const ;

TColor getcolor() const ;

int getfontstyle() const ;

void setfontstyle(int style);

void write(ofstream &fs); // метод записи в открытый бинарный поток

void read(ifstream &fs); // метод чтения содержимого из бинарного потока

private:

void SaveString(String s, ofstream &fs); // промежуточный метод сохранения Unicode строки в поток

String LoadString(ifstream &fs); // промежуточный метод чтения Unicode строки из потока

};

Формат данных для записи и чтения.

Все числовые атрибуты сохраняются как целые числа типа int. Для записи и чтения строковых данных используются промежуточные методы. Общий формат хранения строки следующий :

* Длина строки в виде целого числа типа int
* Массив типа wchar\_t равный длине строки. Так как используются строки типа Unicode длина записанной строки в байтах равна удвоенной длине строки.

**4.2.2 Класс StylesCollection**

Класс StylesCollection позволяет хранить множество стилей в виде коллекции.

class StylesCollection {

public:

bool LoadFromFile(String filename); //загрузить из файла

bool SaveToFile(String filename); // сохранить в файл

void FillComboBox(TComboBox \*cb); // заполнить содержимым компонент TComboBox

void AddStyle(Style \*style); // добавить новый стиль в коллекцию

void DeleteStyle(String stylename); // удалить стиль из коллекции

private:

stylesvector styles;

};

stylesvector описана с помощью typedef как:

typedef vector<Style\*> stylesvector;

Как видно множество стилей сохранено в Vector из STL.

В файле коллекция стилей сохранена в виде:

Количество стилей как число типа int

Последовательность структур которые читаются методом read класса Style.

**4.2.3 Класс HTMLDocument**

Хранит HTML документ, позволяет наблюдать за изменением компонентов, обновлять компоненты при их изменении. Сохранять и загружать документ, контролировать изменения документов.

class HTMLDocument {

public:

String FileName; // имя файла

String html; // содержимое документа

HTMLDocument(TRichEdit \*rche, BrowserSys \*browser); //конструктор

void OpenFile(const String &filename); // метод загрузки из файла

void SaveFile(const String &filename); // метод записи в файл

bool changed(); // возвращает значение, сигнализирующее об измение документа после последней загрузки или сохранения

void setchanged(bool flag); // метод позволяющий изменять этот флаг

void Update(const String &html); // метод вызывающийся при изменеиии контента в RichEdit

};

Сохранение и загрузка документа осуществлена с с помощью встроенного в

C++ Builder класса.

**4.2.4 Класс BrowserSys**

Класс-оберка над низкоуровневым интерфейсом WebBrowser.

class BrowserSys {

public:

BrowserSys(TCppWebBrowser \*wb, TPopupMenu \*popupmenu);

// конструктор – принимает указатель на компонент TCppWebBrowser и меню для него TPopupMenu

~BrowserSys(); // деструкор

bool InitInterfaces(); // метод инициализирующий внутренние интерфейсы

void EditMode(bool on = true); // метод перевода браузера в режим реактирования

bool CanPaste(); // предикат указывающий на возможность вставки из буфера обмена

bool CanCopy(); // возможность копирования в буфер

bool CanCut(); // возможно вырезать в буфер

bool CanRedo(); // возможность повторить отмененное действие

bool CanUndo(); // возможность отменить действие

void Bold(); // установить полужирный стиль начертания для выделенного текта

void Italic(); // установить курсивное начертание

void Paste(); //вставка текста из буфера в позицию курсора

void Copy(); // копирование в буфер

void Cut(); // вырезать в буфер

void UnderLine(); // утсановить подчеркнутое начертание

void InsertList(); // вставить список

void HyperLink(); //вставить гиперссылку

void InsertImage();// вставить изображение

void Undo(); // отменить последнее действие

void Redo();// повторить последнее действие

void SetSize(int size);// установить размер выделенного текта (1..7)

void SetFont(const String &font);// установить шрифт

void SetColor(int color);// установить цвет

int GetSize(IHTMLTxtRange\* TextRange); // получить размер выделенного текста

String GetFont(IHTMLTxtRange\* TextRange); // получить шрифт

int GetColor(IHTMLTxtRange\* TextRange); // получить цвет

bool isItalic(IHTMLTxtRange\* TextRange); // подчеркнут ли текст

bool isBold(IHTMLTxtRange\* TextRange);

bool isUnderline(IHTMLTxtRange\* TextRange);

IHTMLTxtRange\* TxtRange(); ///получить текстовую область

void SetStyle(Style \*style);// установить форматирование текста в соответствии с данными из объекта Style

String GetText(); // получить текст

void SetText(String s); // установить текст

void OpenFile(String filename); // загрузить документ из файла напрямую

};

**5 ОПИСАНИЕ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ**

**Блок схемы алгоритмов**

Запись стилей в файл

Начало

Открыть файл

i=0

Запись количества стилей в файл

Конец

Закрытие файла

Запись стиля

I++

i < count

Нет

Рисунок 5.1

Удаление стиля из коллекции

Начало

stylesvector::iterator i

Style \*st

I != styles.end()

st = \*i

St->getstylename() == stylename

styles.erase(i)

delete st

Конец

st = \*i

i = styles.begin()

Нет

Нет

Рисунок 5.2

Реализация алгоритма удаления стиля TForm1::acDeleteStyleExecute(TObject \*Sender)

Шаг 1 Начало

Шаг 2 Определить номер выбранного элемента в cbStyle

Шаг 3 С помощью операции индексации получить строку с текстом выбранного пункта

Шаг 4 Проверка равен ли первый символ строки ‘@’

Шаг 5 Если не равен то вывести сообщение о невозможности удалить стиль и перейти к шагу11

Шаг 6 Удалить выбранный стиль из коллекции пользовательских стилей styles

Шаг 7 Очистить cbStyle

Шаг 8 Заполнить cbStyle стандартными стилями

Шаг 9 Заполнить cbStyle пользовательскими стилями

Шаг 10 Выбрать первый элемент в cbStyle

Шаг 11 Конец

Реализация алгоритма сохранения файла void \_\_fastcall TForm1::acSaveFileAsExecute(TObject \*Sender)

Шаг 1 Начало

Шаг 2 Получить указатель на активную MDI форму

Шаг 3 Если указатель пустой то переход к пункту 10

Шаг 4 Установить имя файла по имени файла из документа

Шаг 5 Установить текущую директорию по имени файла из документа

Шаг 6 Если Диалог сохранения файла выполнился успешно выполнить, иначе переход к шагу 10

Шаг 7 Установить имя файла документа в соответствии с именем файла

Шаг 8 Установить заголовок активной формы

Шаг 9 Выполнить сохранение документа под заданным именем

Шаг 10 Конец

Реализация алгоритма обработки запроса на закрытие формы

Шаг 1 Начало

Шаг 2 Проверка, изменился ли документ

Шаг3 Если документ не изменился переход к шагу 12

Шаг 4 Вывести запрос на закрытие документа

Шаг 5 Проверить. Какая кнопка была нажата

Шаг 6 Если была нажата кнопка “No” то установить CanClose – true. Перейти к шагу 13

Шаг 7 Иначе. Установить CanClose false

Шаг 8 Если не была нажата кнопка “Yes” переход к шагу 13

Шаг 9 Получить указатель на главную форму

Шаг 10 Сделать текущий документ активным

Шаг 11 Вызвать диалог сохранения файла

Шаг 12 Установить CanClose = true

Шаг 13 Конец

**6 Руководство пользователя**

Программный продукт состоит из следующих файлов

Editor.exe исполняемый файл программы

Styles.dat – файл хранящий пользовательские стили

Интерфейс программы представлен на рисунке 6.1

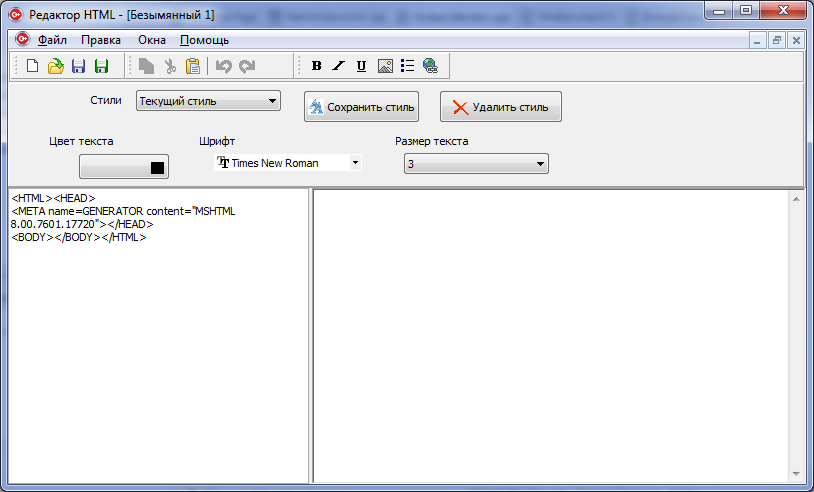


Рисунок 6.1 – Интерфейс программы

Для открытия существующего документа нужно выбрать пункт меню Файл – Открыть файл

Редактирование документа осуществляется одновременно в двух окнах:

В левом окне на уровне HTML кода;

В правом окне визуально.

**ЗАКЛЮЧЕНИЕ**

В результате проведенной работы было разработано программное средство для редактирования HTML документов.

В ходе создания были изучены принципы создания графического интерфейса в среде C++ Builder изучены принципы работы с COM интерфейсами.

Программный продукт полностью удовлетворяет поставленной задаче.

Программа проста в использовании и не перегружена пользовательским интерфейсом.

В дальнейшем, возможно, улучшить программу, организовав добавление HTML элементов, элементов html форм. Возможность работы с разными кодировками.

**ЛИТЕРАТУРА**

1. Г. Шилдт «Программирование на С++»
2. MSDN (msdn.microsoft.com).
3. А. Архангельский «Программирование в C++ Builder 6»
4. RAD Studio XE2 Online Help (http://docwiki.embarcadero.com/RADStudio/en/)