**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**«БАРАНОВИЧСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**Факультет \_инженерный\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Кафедра \_информационных технологий и физико-математических**

**дисциплин\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дата регистрации работы в деканате \_\_\_\_\_\_\_\_\_**

**Дата регистрации работы на кафедре \_\_\_\_\_\_\_\_\_**

**Отметка о допуске к защите \_\_\_\_\_\_\_\_\_**

**Оценка за защиту**  **\_\_\_\_\_\_\_\_\_**

**КУРСОВАЯ РАБОТА**

**по дисциплине** Основы алгоритмизации и программирования

**Тема: «**Разработка диалоговых программ средствами С++ Builder**»**

**Исполнитель:**

**\_**Инженерный, 1курс, ИСТ-11**\_\_\_\_**

**студент (факультет, курс, группа)**

**\_Шмыгин Никита Сергеевич\_**

**фамилия, имя, отчество**

**Руководитель:**

Старший преподаватель кафедры Информационных систем и

физико-математических дисциплин. Магистр педагогических наук**\_\_\_\_\_\_\_\_**

**ученое звание, ученая степень, должность**

**\_**Раковцы Галина Махайловна**\_\_\_\_\_\_\_\_**

**фамилия, имя, отчество**

**Барановичи 2022 год**

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**«БАРАНОВИЧСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

РЕЦЕНЗИЯ

на курсовую работу

(регистрационный №\_\_\_\_\_)

Студента

\_\_\_\_\_\_\_\_\_\_Шмыгина Никиты Сергеевича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

Факультет \_\_\_\_\_\_\_\_\_\_\_\_\_\_ИНЖЕНЕРНЫЙ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Курс \_\_\_\_\_1\_\_\_\_\_\_\_\_\_\_\_

Дисциплина

\_\_Основы алгоритмизации и программирования\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рецензент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Раковцы Галина Михайловна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

Дата получения к/р для рецензирования \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата возвращения к/р после рецензирования \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись преподавателя-рецензента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Текст рецензии:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 

# **РЕФЕРАТ**

Курсовая работа содержит: 28 страниц, 16 рисунков, 9 источников, 1 приложение.

ПРИЛОЖЕНИЕ, ФАЙЛЫ, МАССИВЫ, С++.

Предметом исследования является применение среды программирования C++ Builder для создания программы, которая находит букву, с которой начинается больше всего слов в тексте.

Объект исследования: информационный поиск данных.

При выполнении работы применён метод: изучение научно-методической литературы, проектов.

В ходе деятельности проведены следующие исследования и разработки: работа с визуальными компонентами C++ Builder, запись и вывод в файл, осуществление алгоритма программы «Поиск буквы, с которой начинается больше всего слов в тексте».

Сферой возможного практического применения является обработка и поиск информации в текстах.

Автор подтверждает, что нижеприведенный в работе расчетно-аналитический материал правильно и объективно отражает состояние исследуемого процесса, а все заимствованные из литературных и других источников теоретические, методологические и методические положения и концепции сопровождаются ссылками на их авторов.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**СОДЕРЖАНИЕ**

[**РЕФЕРАТ** 3](#_Toc105710643)

[**ВВЕДЕНИЕ** 5](#_Toc105710644)

[1 **ТЕОРЕТИЧЕСКАЯ ЧАСТЬ** 6](#_Toc105710645)

[1.1 Постановка задачи 6](#_Toc105710646)

[1.2 Язык программирования С++ 6](#_Toc105710647)

[1.3 Теоретические аспекты предметной области 7](#_Toc105710648)

[1.4 Алгоритм решения задачи 9](#_Toc105710649)

[1.5 Блок-схема алгоритма 9](#_Toc105710650)

[2. **ПРАКТИЧЕСКАЯ ЧАСТЬ** 13](#_Toc105710651)

[2.1 Описание объектов, свойств, методов 13](#_Toc105710652)

[2.2 Функции и переменные используемые в программе 18](#_Toc105710653)

[2.3 Формы и основные модули текста программы. 19](#_Toc105710654)

[2.4 Результаты тестирования программного продукта 22](#_Toc105710655)

[2.5 Анализ результатов 26](#_Toc105710656)

[**ЗАКЛЮЧЕНИЕ** 27](#_Toc105710657)

[**CПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ** 28](#_Toc105710658)

[**ПРИЛОЖЕНИЕ А** 29](#_Toc105710659)

# **ВВЕДЕНИЕ**

В настоящее время ЭВМ широко применяются во многих отраслях деятельности человека. Ни одна компания не может обойтись в своей работе без применения автоматизации, которая с успехом реализует рутинную работу, выполнявшуюся ранее в ручную, повышая эффективность работы любой компании. Для автоматизации используют языки программирования высокого уровня, позволяющие записывать алгоритмы решаемых задач в довольно естественном виде.

Информационный поиск — большая междисциплинарная область науки, стоящая на пересечении [когнитивной психологии](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B3%D0%BD%D0%B8%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D0%BF%D1%81%D0%B8%D1%85%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F), [информатики](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0), [лингвистики](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%B3%D0%B2%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0), [семиотики](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B8%D0%BE%D1%82%D0%B8%D0%BA%D0%B0_(%D1%8F%D0%B7%D1%8B%D0%BA%D0%BE%D0%B7%D0%BD%D0%B0%D0%BD%D0%B8%D0%B5)) и [библиотечного дела](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0). Поиск информации — процесс выявления в массиве информации записей, удовлетворяющих заранее определенному условию поиска или запросу.

Целью курсовой работы является обобщить знания по управляющим операторам языка С/С++; получить практические навыки работы с файлами, овладеть практическими навыками в организации ввода/вывода значений комбинированных типов данных, получить практические навыки программирования задач поиска данных в строках файлов в среде С++Builder.

Задачи курсовой работы:

1) изучение необходимых теоретических сведений о поиске и обработке данных текстовых файлов;

2) построение и описание блок-схем алгоритмов;

3) создание программы, ее отладка и тестирование.

Средства реализации программного кода:

* методы и методики алгоритмирования;
* язык программирования С++;
* отладчик С++Builder.

# 1 **ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

## Постановка задачи

Необходимо создать программу, которая будет обрабатывать исходный файл, в котором задан текст на русском языке. Исходный текст должен заканчиваться точкой ('!', '?', '…'). Составить программу для нахождения буквы, с которой начинается больше всего слов в тексте (прописную и заглавную буквы считать одинаковыми). В файле результатов должен быть исходный текст и слова, начинающиеся с заданной буквы (с новой строки и с порядковым номером). Длина текста — не более NL строк, длина строки — не более NS символов, длина слова — не более NW символов.

* 1. **Язык программирования С++**

Для разработки данного приложения был выбран язык программирования С++.

Язык программирования С++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений.

На сегодняшний день С++ является одним из самых популярных и распространенных языков. С++ является мощным языком, унаследовав от Си богатые возможности по работе с памятью. Поэтому нередко С++ находит свое применение в системном программировании, в частности, при создании операционных систем, драйверов, различных утилит, антивирусов и т.д. К слову сказать, ОС Windows большей частью написана на С++. Но только системным программированием применение данного языка не ограничивается. С++ можно использовать в программах любого уровня, где важны скорость работы и производительность. Нередко он применяется для создания графических приложений, различных

прикладных программ. Также особенно часто его используют для создания игр с богатой насыщенной визуализацией. Кроме того, в последнее время набирает ход мобильное направление, где С++ тоже нашел свое применение. И даже в веб-разработке также можно использовать С++ для создания веб-приложений или каких-то вспомогательных сервисов, которые обслуживают веб-приложения. В общем С++ - язык широкого пользования, на котором можно создавать практически любые виды программ.

С++ является компилируемым языком, а это значит, что компилятор транслирует исходный код на С++ в исполняемый файл, который содержит набор машинных инструкций. Но разные платформы имеют свои особенности, поэтому скомпилированные программы нельзя просто перенести с одной платформы на другую и там уже запустить. Однако на уровне исходного кода программы на С++ по большей степени обладают переносимостью, если не используются какие-то специфичные для текущей ос функции. А наличие компиляторов, библиотек и инструментов разработки почти под все распространенные платформы позволяет компилировать один и тот же исходный код на С++ в приложения под эти платформы.

Язык C++ позволяет писать приложения в объектно-ориентированном стиле, представляя программу как совокупность взаимодействующих между собой классов и объектов. Что упрощает создание крупных приложений.

## 1.3 Теоретические аспекты предметной области

В данной курсовой работе особое внимание уделяется работе с файлами и строками, а также организация ввода/вывода значений комбинированных типов данных.

Текстовыми называются файлы, состоящие из любых символов. Они организуются по строкам, каждая из которых заканчивается символом «конца строки». Конец самого файла обозначается символом «конца

файла». При записи информации в текстовый файл, просмотреть который можно с помощью любого текстового редактора, все данные преобразуются к символьному типу и хранятся в символьном виде.

Для работы со строками в C++ Builder реализованы классы: String, AnsiString, TStringList.

Символ в большинстве компьютерных языков - это тип данных, предназначенный для хранения одной буквы или символа, поскольку он представлен кодировкой символов, используемой программой или операционной системой.

Поставленная задача будет реализована средствами среды С++Builder.

C++ Builder– это полноценный расширяемый программный продукт с удобной разработкой приложений RAD. В C++ Builder внедрена оптимизированная среда программирования IDE, которая позволяет разрабатывать на языке C++ различного направления программное обеспечение.

C++ Builder – обширный комплекс с такими библиотеками как STL, MFC, VCL, CLX и др. Кроме этого имеется компилятор, редактор, отладчик кода и множество других полезных для разработки компонентов. Процесс разработки приложений схож с Delphi, поэтому большинство компонентов, которые разрабатывались под Delphi, можно интегрировать в C++ Builder без каких-либо модификации и бубнов. В C++ Builder большая часть компонентов перешла из Delphi. Благодаря этому, написанный код на языке Delphi, с минимальными знаниями можно преобразовать в C++ Builder.

В C++ Builder есть огромное количество инструментов, которые позволяют упростить разработку визуальных приложений с помощью drag-and-drop компонентов и встроенного редактора интерфейса WYSIWYG.

## 1.4 Алгоритм решения задачи

Разработка алгоритма решения задачи (алгоритмизация) - один из самых сложных и ответственных этапов. Он включает формализацию процесса обра­ботки данных и представление его в форме, позволяющей упростить дальней­шее кодирование алгоритма, т. е. его запись на одном из алгоритмических языков. Алгоритмизация включает выделение предполагаемых этапов процесса обработки данных и запись их в определенной форме и последовательности, например в виде схем алгоритмов.

Алгоритм - это точное предписание по выполнению некоторого процесса обработки данных, который через разумное конечное число шагов приводит к решению задачи данного типа для любых допустимых вариантов исходных данных.

При запуске программы происходит открытие исходного файла. Далее следует чтение из файла в строку данных. Полученная строка, делится на массив слов. Осуществляется поиск буквы, с которой начинается больше всего слов в тексте. Далее следует вывод слов с найденной буквой. Последний шаг – запись в файл считаного текста и слов.

## 1.5 Блок-схема алгоритма

Проанализировав поставленную задачу, структура программы может быть представлена следующим образом (рисунок 1.1):

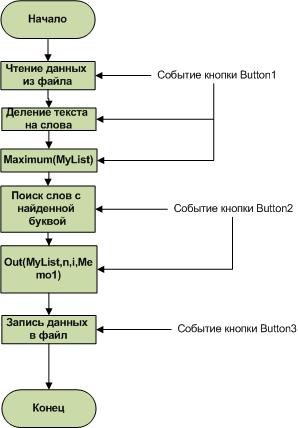


Рисунок 1.1 – Структура программы

На рисунке 1.2 представлена блок-схема алгоритма функции Maximum(TStringList \*MyList) поиска буквы, с которой начинается больше всего слов в тексте.

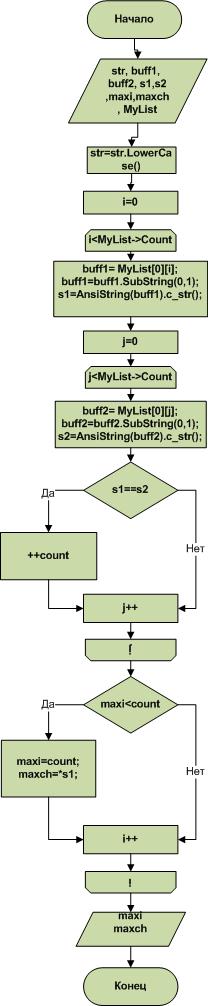


Рисунок 1.2 – Алгоритм поиска буквы

На рисунке 1.3 представлена блок-схема алгоритма поиска слов с найденной буквой и вывод результата в Memo2.

# C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Поиск слова.jpg

Рисунок 1.3 – Алгоритм поиска слова

# 2. **ПРАКТИЧЕСКАЯ ЧАСТЬ**

## 2.1 Описание объектов, свойств, методов

**Компонент TButton.** Необходим для выполнения какого-либо действия по нажатию. Основные свойства представлены в таблице 2.1

Таблица 2.1— Свойства компонента Button

|  |  |
| --- | --- |
| **Свойство** | **Назначение** |
| **Align** | Позициякомпонента.   **alBottom** – Компонентрастягиваетсяпонизуформы    **alClient** – Компонентрастягиваетсяповсейформе    **alCustom** – Определяется **CustomAlignInsertBefore** и**CustomAlignPosition**.    **alLeft** – Компонент растягивается по левому краю формы    **alNone** – Устанавливается при разработке, по умолчанию.     **alRight** – Компонент растягивается по правому краю формы     **alTop** – Компонент растягивается по верху формы |
| **Aligment** | Выравнивание текста     **taRightJustify** – По правому краю     **taLeftJustify** – По левому краю     **taCenter** – По центру |
| **Caption** | Надпись на компоненте. |
| **Color** | Цвет компонента. |
| **Height** | Высота компонента. |
| **Left** | Отступ от левого края формы или экрана. |
| **Height** | Высота компонента. |
| **Left** | Отступ от левого края формы или экрана. |

Продолжение таблицы 2.1

|  |  |
| --- | --- |
| **Constraints** | Максимальные и минимальные размеры компонента     **MaxHeight** – Максимальная высота компонента     **MaxWidth** – Максимальная ширина компонента     **MinHeight** – Минимальная высота компонента     **MinWidth** – Минимальная ширина компонента |
| **Cursor** | Тип курсора, при наведении его на компонент. |
| **Enabled** | Если это свойство установлено в **false**, то компонент становится бледнее и по нему уже нельзя щелкнуть клавишей мыши (можно, но действие в обработчике щелчка не произойдет) или ввести/скопировать какие-либо данные. |
| **Font** | Шрифт компонента.     **Charset** – Набор символов шрифта.     **Color** – Цвет текста     **Height** – Высота текста     **Name** – Имя шрифта     **Pitch** – Шаг шрифта         **fpDefault** – По умолчание, определяется шрифтом.       **fpFixed** – Фиксированный шаг, все символы имеют одинаковую ширину.         **fpVariable** – Устанавливается в переменной. Символы шрифта могут иметь разную ширину.      **Size** – Размер шрифта      **Style** – Стиль шрифта         **fsBold** – Жирный         **fsItalic** – Курсив         **fsUnderline** – Подчеркнутый         **fsStrikeOut** – Зачеркнутый |

Продолжение таблицы 2.1

|  |  |
| --- | --- |
| **Name** | Имя компонента, через которое вы будете обращаться в программе. Рекомендуется заменять на более осмысленное название, чтобы потом не возникало путаницы. |
| **ParentColor** | Наследования характеристик родителя, как правило формы, в данном случае, параметров цвета. |
| **ParentFont** | Наследования характеристик родителя, как правило формы. |
| **ParentShowHint** | Наследования характеристик родителя, как правило формы, в данном случае, разрешение на отображение всплывающей подсказки. |
| **PopupMenu** | В данном свойстве можно выбрать. нужное **PopupMenu** для компонента (если вы его создавали). |
| **ShowHint** | Определяет показывать или не показывать всплывающую подсказку,**true** – показывать. |
| **Top** | Отступ от верхнего края формы или экрана. |
| **Transparent** | Прозрачность, установите в **true**, чтобы было видно другие компоненты через фон данного. |
| **Visible** | Если установлено **true**, то компонент становится невидимым. |
| **Width** | Ширина компонента. |
| **Font** | Шрифт, используемый для отображения текста. Уточняющие свойства определяют шрифт (Name), размер (size), стиль (style) и цвет символов (Color). |
| **Visible** | Позволяет скрыть текст (false) или сделать его видимым (true). |

**Компонент Memo** - компонент для отображения простого текста (как в программе "Блокнот").

Основные свойства представлены в таблице 2.2.

Таблица 2.2 — Свойства компонента Memo

|  |  |
| --- | --- |
| **Свойство** | **Значение** |
| **Aligment** | Выравнивание теста внутри **Memo**. Если равно **taLeftJustify**, то слева, **taRightJustify** - справа, **taCenter** - по центру. |
| **Lines** | Здесь можно удалить или заменить надпись "Memo1". Строки, можно задать программно, а можно сразу, щелкнув по "...".  Добавление программно: **Memo1->Lines->Add("Ваша строка");** |
| **MaxLenght** | Максимальная длина количества вводимых символов, по умолчанию равно 0, то есть не ограничено. |
| **ReadOnly** | Если установлено **true**, то запрещает изменение текста пользователем, но возможно выделение и копирование. |
| **ScrollBars** | Определяет полосу прокрутки, **ssNone** - отсутствует, **ssHorizontal** - горизонтальная, **ssVertical**- вертикальная, **ssBoth** - присутствует и горизонтальная и вертикальная. |
| **WordWrap** | Авто перенос текста по словам при достижении границы, если установлено в **true**. |

**Компонент OpenDialog -** компонент для открытия файлов. **Компонент SaveDialog** **-** компонент для сохранения файлов.

Основные свойства представлены в таблице 2.3.

Таблица 2.3 — Свойства компонентов OpenDialog, SaveDialog

|  |  |
| --- | --- |
| **Свойство** | **Значение** |
| **Ctl3D** | Свойство, необходимое для обратной совместимости. Не имеет значение начиная с Windows XP. |
| **DefaultExt** | Расширение файла по умолчанию. Точку указывать не надо. |
| **FileName** | Данное свойство возвращает имя и полный путь к каталогу недавно выбранных файлов. Для того, чтобы имя файла по умолчанию отображалось в окне редактирования диалога, нужно присвоить значение файла в **Object Inspector** или в коде программы. Программные изменения в **FileName** иметь никакого эффекта, пока диалоговое активно. |
| **Filter** | Диалоговое окно выбора включает в себя выпадающий список типов файлов. Когда пользователь выбирает тип файла из списка, только файлы выбранного типа отображаются в диалоговом окне.  Для настройки маски файлов во время разработки нажмите на кнопку с многоточием знаки (...) справа от фильтра в **Object Inspector**. Это откроет редактор фильтров. В левой колонке редактора фильтра, в **Filter Name** введите краткое описание каждого типа файлов, которые будут доступны во время работы программы. В правой колонке находятся маски фиотров, соответствующие каждому описанию. Например, описание "Текстовые файлы" может появиться слева от маски "\*. TXT", а описание "Файлы jpeg" может появиться слева от маски "\*.jpeg". Программное задание фильтров:  **OpenDialog1-> Filter = "Файлы  (\*jpeg.) | \*JPEG.»;**  Несколько фильтров должны быть разделены вертикальными линиями. Например:  **OpenDialog1->Filter = "Text files (\*.txt)|\*.TXT|Pascal files (\*.pas)|\*.PAS";**  Чтобы включить несколько масок в одном фильтре, отделить маски с запятой. Это работает как в**Object Inspector**, так и в коде программы. Например: |

Продолжение таблицы 2.3

|  |  |
| --- | --- |
|  | **OpenDialog1->Filter = "Pascal files|\*.PAS;\*.DPK;\*.DPR";** Если значение фильтру не присвоено, диалоговое окно отображает все типы файлов. |
| **FilterIndex** | Определяет, какой из типов файлов в фильтре по умолчанию, когда откроется диалоговое окно. Установите 1, чтобы выбрать первый тип файла в списке по умолчанию или установите 2, чтобы выбрать второй тип файла по умолчанию, и так далее. Если значение вне диапазона, то будет выбран первый тип файла, из перечисленных в фильтре. |
| **InitialDir** | Определяет директорию по умолчанию, которая будет отображаться в диалоговом окне выбора, когда оно откроется. |
| **Name** | Отвечает за имя компонента. Для удобства его можно сократить, например - **OPD1**. |
| **Options** | Используется для настройки параметров и внешнего вида диалога. По умолчанию все эти опции выключены |
| **OptionsEx** | Описания, которые определяют внешний вид и поведение выбора файлов в диалоге. |
| **Tag** | Переменная, выделенная для нужд программиста. |
| **Title** | Используется для задание текста, который отображается заголовке диалога. Если значение не присвоено, название имеет значение "**Open**". |

**2.2 Функции и переменные используемые в программе**

Подпрограммы, используемые в основной программе, приведены в таблице 2.4.

Таблица 2.4 - Функции

|  |  |
| --- | --- |
| Функция | Назначение |
| Maximum(TStringList \*MyList) | Поиск буквы, с которой начинается больше всего слов в тексте |
| Out(TStringList \* MyList,int n,int i,TMemo \*Memo1) | Вывод в компоненту Memo найденных слов |

Переменные, используемые в основной программе, приведены в таблице 2.5.

Таблица 2.5 - Переменные

|  |  |
| --- | --- |
| Переменная | Назначение |
| AnsiString TX, FileNameRez; | Переменные с именем файла для чтения и записи |
| FILE\* Text\_TX; | Файловая переменная |
| String str; | Строка с текстом |
| int count | Количество повторений букв в тексте |
| int maxi | Индекс буквы, которая повторяется максимальное число раз |
| char maxch | Буква, которая повторяется максимальное число раз |
| String buff1, buff2, buff | Временные переменные, для хранения букв |
| char \* s1,s2,s3 | Временные переменные, для хранения букв, как символа |
| TStringList \* ptStrintList | Массив со строками текста из файла |
| TStringList \* MyList | Массив со строками слов |
| int n=0 | Нумерация итоговых слов |
| String s | Одно слово из массива |

## 2.3 Формы и основные модули текста программы.

Разработанное приложение является оконным.

Для корректной работы программы необходим файл с исходным текстом. На рисунке 2.1 представлен исходный файл с текстом.

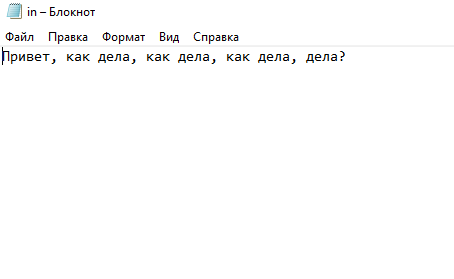


Рисунок 2.1 – Исходный файл

На рисунке 2.2 представлен интерфейс пользователя.

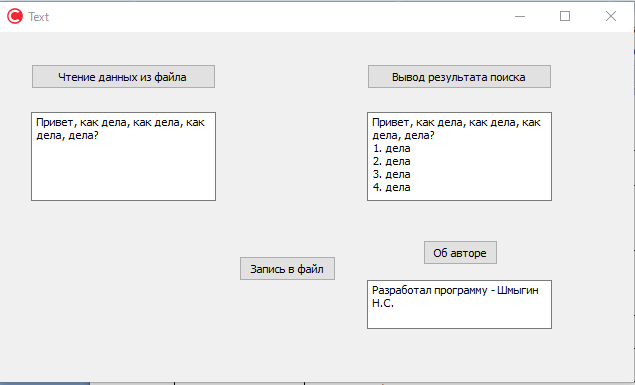


Рисунок 2.2 – Интерфейс пользователя

На рисунке 2.3 представлен файл с результатом работы программы.

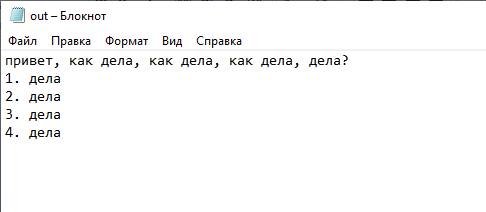


Рисунок 2.3 – Файл результатов

Рассмотрим исходный код поиска слов с найденной буквой в начале слова и вывода результата в Memo2.

int n=0;

String buff;

for(int i=0; i<ptStrintList->Count;i++)

Memo2->Lines->Add(ptStrintList[0][i]);

for(int i=0; i<MyList->Count;i++){

buff= MyList[0][i];

buff=buff.SubString(0,1);

char\* s3 = AnsiString(buff).c\_str();

if(\*s3 == maxch){

n++;

Out(MyList,n,i,Memo2);

}

}

Задаем начальное значение для нумерации слов n=0. В цикле добавляем в компоненту Memo2 начальный текст. Реализовываем цикл поиска слов. Слова текста хранятся в массиве MyList. Берем элемент массива и записываем в отдельную переменную buff. Переопределяем переменную значением первой буквы слова. Меняем тип данных переменной buff и присваиваем значение переменной s3. Если первая буква слова совпадает с найденной буквой, нумерация увеличивается на единицу, вызываем функцию вывода номера слова и слова в Memo2.

## 2.4 Результаты тестирования программного продукта

Открываем окно, для выбора исходного файла. На рисунке 2.4 представлена работа компоненты OpenDialog.

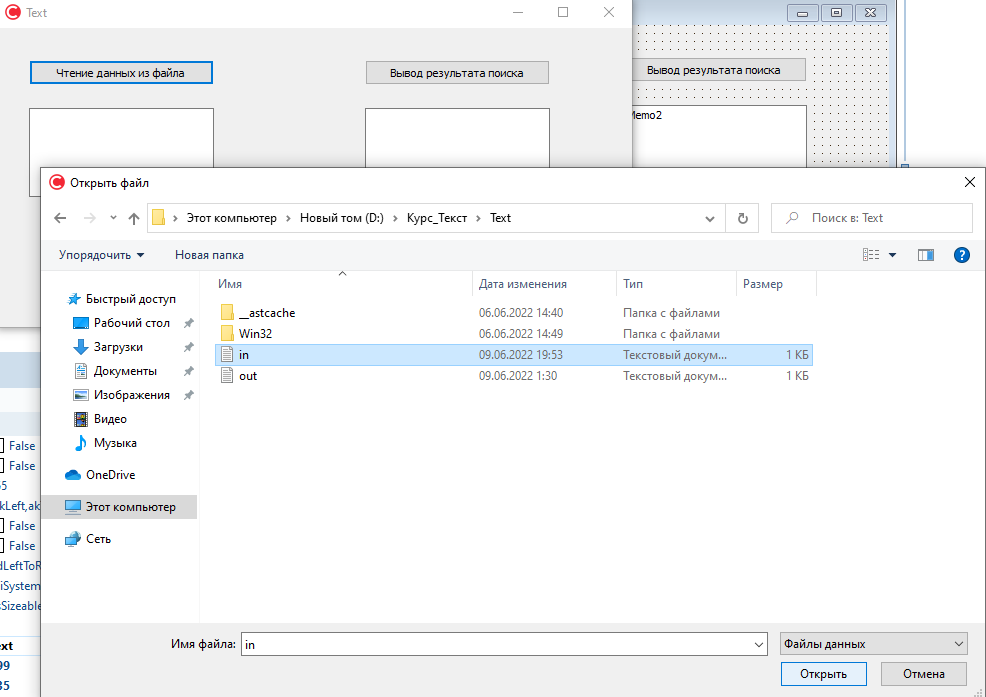


Рисунок 2.4 – Выбор исходного файла

Содержимое файла отображено на рисунке 2.5.

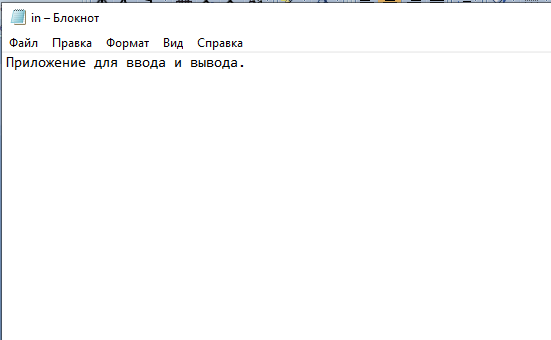


Рисунок 2.5 – Исходный файл

Вывод данных из файла в Memo1 представлено на рисунке 2.6.

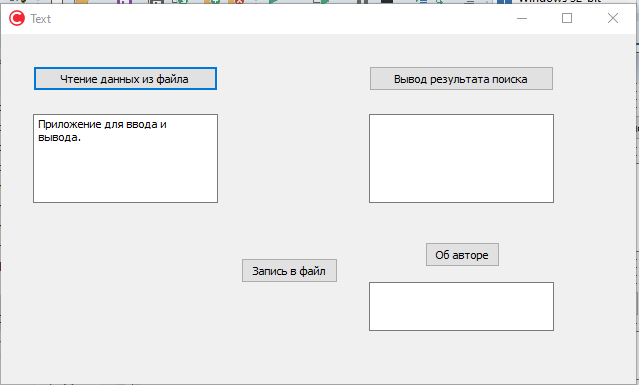


Рисунок 2.6 – Вывод данных из файла в Memo1.

На рисунке 2.7 отображается результат поиска.

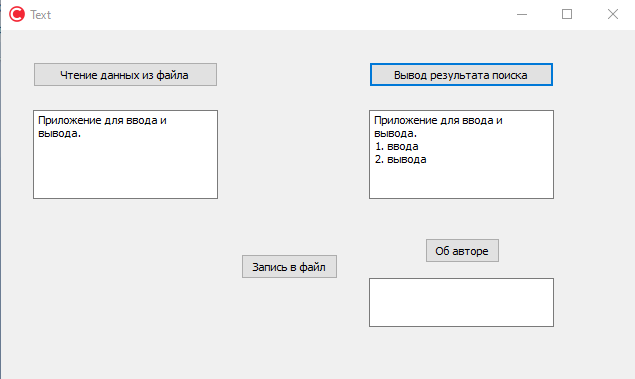


Рисунок 2.7 – Результат поиска

На рисунке 2.8 представлена работа компоненты SaveDialog .

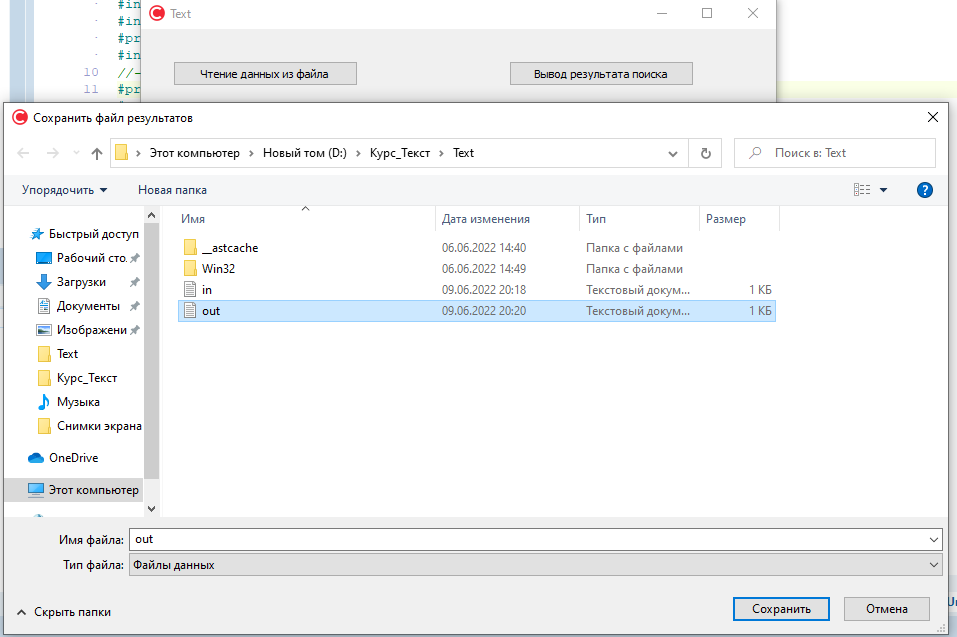


Рисунок 2.8 – Создание файла для записи результатов

Содержимое файла отображено на рисунке 2.9.

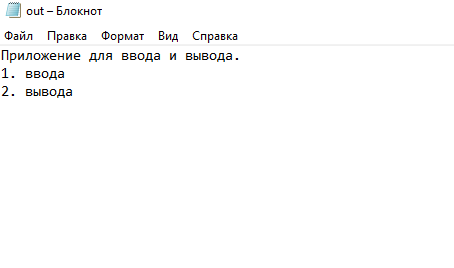


Рисунок 2.9 – Файл результатов

Проверим работу кнопки «Об авторе». На рисунке 2.10 представлен результат.

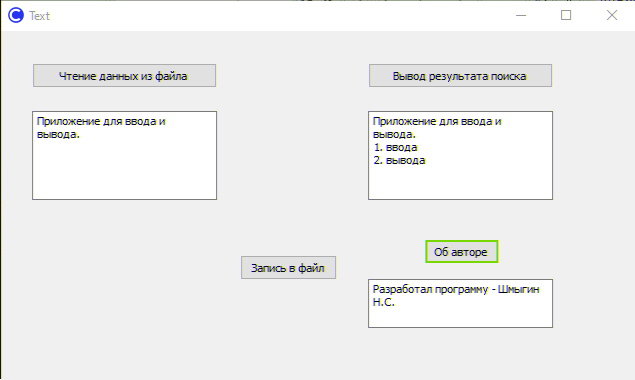


Рисунок 2.10 – Вывод информации об авторе

Видим, что программа отработала корректно: найдено два слова, которые начинаются с одинаковой буквы.

Зададим больше слов, с повторяющимися первыми буквами, а также проверим считается ли прописная и заглавная буквы одинаковыми. На рисунке 2.11 представлен исходный файл.

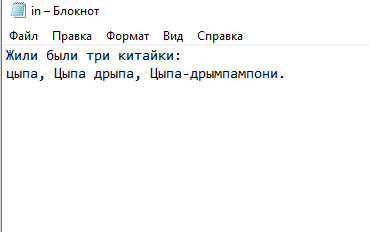
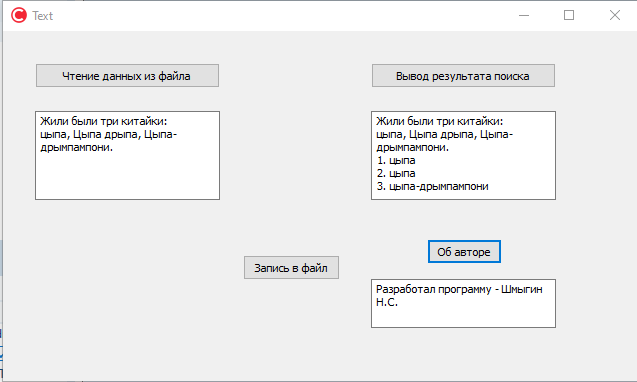


Рисунок 2.11 – Исходный текст

Вывод данных из файла и результат поиска отображены на рисунке 2.12.

Рисунок 2.12 – Вывод данных из файла и результат поиска

Содержимое файла с результатами отображено на рисунке 2.13.

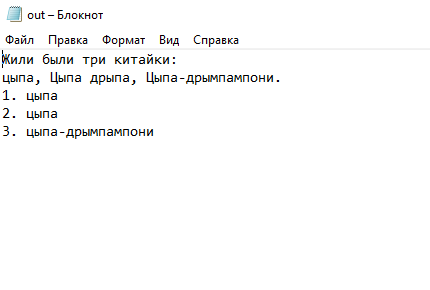


Рисунок 2.13 – Файл с результатами

## 2.5 Анализ результатов

В ходе тестирования приложения ошибок выявлено не было. Программа работает корректно.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы были получены практические навыки работы с файлами, применены практические навыки в организации ввода/вывода значений строк, были получены практические навыки программирования задач с использованием массивов строк и файлов в среде С++Builder.

Проведенное тестирование программы позволяет сделать вывод о работоспособности программы и ее соответствии заданию. Выявленные недостатки предполагают дальнейшую работу над совершенствованием программы, что предоставляет начинающему программисту обширное поле деятельности для повышения знаний и совершенствовании навыков программирования в данной среде, освоении новых аспектов программирования в современной среде.

Данная программа может быть использована, например, в библиотеке, для поиска необходимой информации.

# **CПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Campbell Parallel Programming with Microsoft® Visual C++® / Campbell. - Москва: **Гостехиздат**, 2011. - **784** c.  
2. Альфред, В. Ахо Компиляторы. Принципы, технологии и инструментарий / Альфред В. Ахо и др. - Москва: **Высшая школа**, 2015. - **882** c.

3. Мешков, А. Visual C++ и MFC / А. Мешков, Ю. Тихомиров. - М.: БХВ-Петербург, **2013**. - **546** c.

4. Положение о курсовых работах (проектах) №01-01; МО Республики Беларусь, БарГУ. – Барановичи, 2007.

5. Паневик В.В. и др. Общие требования, порядок выполнения и правила оформления студенческих работ и магистерских диссертаций. /В.В. Паневик, Л.А. Лобан, С.В. Некраха. – Мн.: БГТЭУ, 2004.

6. М.А. Зильберглейт, Л.И. Петрова. Методика и техника подготовки курсовых и дипломных работ. – Мн.: «Беларускаянавука», 2003.

7. Курсовые и дипломные работы: от выбора темы до защиты: Справочное пособие/ Авт.-сост. И.Н. Кузнецов. – Мн.: «Мисанта», 2003.

# **ПРИЛОЖЕНИЕ А**

Код программы

//---------------------------------------------------------------------------

#include <vcl.h>

#include<io.h>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

#define NL 3

#define NS 20

#define NW 20

TForm1 \*Form1;

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

Memo1->Clear();

Memo2->Clear();

Memo3->Clear();

}

AnsiString TX, FileNameRez;

FILE\* Text\_TX;

String str;

int count=0;

int maxi=0;

char maxch;

TStringList \* MyList=new TStringList();

TStringList \* ptStrintList=new TStringList();

void Out(TStringList \* MyList,int n,int i,TMemo \*Memo1){

String s = MyList[0][i];

if(s.Pos('.')){

int index = s.Pos('.')-1;

s=s.SubString(0, index);

Memo1->Lines->Add(IntToStr(n)+". "+s);

}

else if(s.Pos('!')){

int index = s.Pos('!')-1;

s=s.SubString(0, index);

Memo1->Lines->Add(IntToStr(n)+". "+s);

}

else if(s.Pos(',')){

int index = s.Pos(',')-1;

s=s.SubString(0, index);

Memo1->Lines->Add(IntToStr(n)+". "+s);

}

else if(s.Pos('?')){

int index = s.Pos('?')-1;

s=s.SubString(0, index);

Memo1->Lines->Add(IntToStr(n)+". "+s);

}

else if(s.Pos(':')){

int index = s.Pos(':')-1;

s=s.SubString(0, index);

Memo1->Lines->Add(IntToStr(n)+". "+s);

}

else

Memo1->Lines->Add(IntToStr(n)+". "+s);

}

void Maximum(TStringList \*MyList){

String buff1, buff2;

for(int i=0; i<MyList->Count;i++){

buff1= MyList[0][i];

buff1=buff1.SubString(0,1);

char \* s1=AnsiString(buff1).c\_str();

for(int j=0;j<MyList->Count;j++){

buff2= MyList[0][j];

buff2=buff2.SubString(0,1);

char \* s2=AnsiString(buff2).c\_str();

if(s1==s2){

count++;

}

}

if(maxi<count){

maxi=count;

maxch=\*s1;

}

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

OpenDialog1->Title="Открыть файл";

if (OpenDialog1->Execute()) {

TX=OpenDialog1->FileName;

if((Text\_TX=fopen(TX.c\_str(),"rb"))==NULL){

ShowMessage("Ошибка открытия файла"); return;

}

Memo1->Lines->LoadFromFile(TX,TEncoding::UTF8);

}

ptStrintList->LoadFromFile(TX,TEncoding::UTF8);

for(int i=0; i<ptStrintList->Count; i++){

str+=ptStrintList[0][i];

}

str=str.LowerCase();

MyList->StrictDelimiter = true;

MyList->Delimiter = ' ';

MyList->DelimitedText=str;

Maximum(MyList);

fclose(Text\_TX);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button3Click(TObject \*Sender)

{

SaveDialog1->Title="Сохранить файл результатов";

if(FileNameRez!=""){

Memo2->Lines->SaveToFile(FileNameRez);

}

else

if (SaveDialog1->Execute()) {

FileNameRez = SaveDialog1->FileName;

Memo2->Lines->SaveToFile(FileNameRez);

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{

int n=0;

String buff;

for(int i=0; i<ptStrintList->Count;i++)

Memo2->Lines->Add(ptStrintList[0][i]);

for(int i=0; i<MyList->Count;i++){

buff= MyList[0][i];

buff=buff.SubString(0,1);

char\* s3 = AnsiString(buff).c\_str();

if(\*s3 == maxch){

n++;

Out(MyList,n,i,Memo2);

}

}

MyList->Free();

ptStrintList->Free();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button4Click(TObject \*Sender)

{

Memo3->Lines->Add("Разработал программу - Шмыгин Н.С.");

}

//---------------------------------------------------------------------------